EXPLOITATION OF LONG-RANGE DEPENDENCE IN INTERNET TRAFFIC FOR RESOURCE AND TRAFFIC MANAGEMENT

BY

GUANGHUI HE

B.E., Tsinghua University, 1993
M.E., Tsinghua University, 1996

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

Abstract

A long-held belief regarding the scaling behavior—self-similar or Long-Range Dependence (LRD) of Internet traffic is that the scaling behavior has adverse impacts on the Internet. In particular, it causes the queue length tend to infinity and hence severe packet drops. The scaling property means much more bursty traffic which consists of concentrated periods of high activity and low activity at a wide range of time scales, and this burstiness adversely affects resource management and degrades the overall Internet performance.

In this thesis we study the scaling behavior of the Internet traffic from a totally different perspective, i.e., we aim to take advantage of the self-similar/LRD property of the Internet traffic for the sake of resource and traffic control. Since self-similar/LRD implies the existence of nontrivial correlation structure at any time scales, accurate prediction can be faithfully achieved and the prediction results can be used to do resource and traffic control. In light of this, first, in order to make a thorough understanding of the scaling behavior, we propose a hierarchical model that has an one-on-one correspondence to the protocols in the protocol hierarchy of IP networks and give insights on how, and to what extent, the user/protocol behavior in each protocol layer contributes to scaling properties. Then based on the understanding, we propose: (1) Predictive Active Queue Management (PAQM) to stabilize the queue lengths at routers based on the prediction of future incoming traffic; (2) TCP with Traffic Prediction (TCP-TP) to fasten the process of reaching the optimal operational point in the congestion control based on the prediction of attainable throughput one or two RTTs ahead in the future; (3) Three theoretically grounded algorithms: prediction, reconstruction and interpolation to do proactive non-intrusive end-to-end measurement. We implement them in both user and kernel level. Real Internet experiments are done based on the user-level implementation; Last, for the techniques of passive Internet traffic sampling, we provide an in-depth analytical study of the sampling techniques for the scaling-behaved Internet traffic.

To overcome the adverse impacts of the scaling property (heavy-tailedness), we propose a Biased Systematic Sampling (BSS) method to capture both the first and second order statistics of the Internet traffic.

To my loved ones

Acknowledgments

This thesis was done under the direction of my advisor, Prof. Jennifer C. Hou. It would not have been possible without her help. Over the years, Prof. Hou has guided, nurtured and supported me through my graduate study. I am especially grateful for the freedom she bears for me to pursue research directions that interest me. I would also like to thank Professors Kihong Park, Lui Sha, Nahrstedt Klara and Rayadurgam Srikant for agreeing to serve on my thesis committee and for their constructive inputs to my thesis research. Special thanks to my former and current fellow graduate students, Yuan Gao, Rong Zheng, Xiao Li, Honghai Zhang for technical discussion and friendship. Lastly, I would like to thank my parents and my sister for their constant support over the years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet layered architecture together with its protocol stacks imposes significant impacts on the characteristics of the Internet traffic it generates. The statistical properties of the Internet traffic, on the other hand, dramatically influences the performances of the Internet. As one of the major objectives of the Internet is to maximize the data throughput, while fulfilling quality-of-service (QoS) requirements imposed by users. Solutions to issues such as how to perform resource provisioning, how to design traffic congestion control policies, and how to route packets to their destinations, heavily depend on a thorough understanding of the stochastic properties of the Internet traffic. Therefore, analyzing the Internet traffic and its use in resource and traffic control plays a critical role in the Internet research and is the focus of the thesis.

Characterizing Internet Traffic Although a long-held paradigm for describing voice and data traffic has been the Markov models (Poisson-alike processes), recent examinations of the traffic in local area networks (LANs) [83] and wide area networks (WANs) [103] have challenged the validity of these models. If traffic followed a Poisson or Markovian arrival process, it would have a characteristic burst length which is smoothed by averaging over a long enough time scale. Rather, measurements of real traffic indicate that significant traffic variance (burstiness) is present at a wide range of time scales. This is termed as the scaling phenomena.

Traffic that is bursty at multiple time scales can be described statistically using the notion of self-similarity, mono-fractals, multi-fractals and power-law or heavy-tailed distributions. Specifically, self-similarity is the property that is associated with one type of fractal—an object whose appearance is unchanged regardless of the scale at which it is viewed. In the context of stochastic

processes, self-similarity is endowed with the following stochastic meaning: when measured at multiple scales, if a process follows the same distribution law (e.g., the probability density function, or pdf), it is called strict self-similar or monofractal process. The process is called second order self-similar if its correlation structure remains unchanged. The self-similar property is characterized mathematically by the $Hurst$ parameter, denoted as H. H is usually a real number in $(0, 1)$. When $H > 0.5$, the process is also called long-range dependence (LRD), which is closely related to power-law or heavy-tailed distributions. If a process has the same value of H at all time scales, it is strictly self-similar or mono-fractal, otherwise, it is called multi-fractal. A multi-fractal process exhibits bursts—extended periods above or below the mean—at a wide range of time scales.

Research in Characterizing Internet traffic Since Leland *et al.* [83] laid the groundwork for understanding the self-similarity/LRD nature of Internet traffic and its impact on network performance, an explosion of research activities has been induced to investigate the multifaceted nature of this phenomenon. Compared with traditional Poisson/Markovian processes, a self-similar process has observable bursts at a wide range of timescales. Thus, it is prone to exhibit long-range dependence, i.e., values at any instant are typically non-negligibly positively correlated with values at all future instants. The long-range dependent feature in network traffic has been observed in several empirical studies [45,81,99], and influences the packet loss and delay behavior in a radically different manner.

The research activities on characterizing the Internet traffic can be categorized into traffic modeling, queuing analysis and traffic control, and resource provisioning. Traffic modeling aims to characterize the Internet traffic with the use of parameterized models. Among all the mathematical models, the models reported in [84,97,101] are proposed to facilitate queuing analysis, while those reported in [5,47,129] aim to capture self-similar burstiness (at multiplexing points in the network) induced by the network protocols and/or architectures. The authors in [44, 75] show that long-range dependence may be generated in diverse ways. The causes of self-similarity are investigated in [131] at the source level, while the authors of [33] cite the distribution of file sizes, the effects of caching and human factors (such as the response time and preference) as possible causes of the self-similarity in WWW traffic. The work in [99] and [47] point out that closed loop protocols such as TCP lead to much richer scaling behaviors than open loop protocols like UDP. The authors of [127]

attribute the self-similarity of TCP traffic to the chaotic nature of its congestion control mechanism. The authors of [52, 59, 117] claim that the retransmission and congestion control mechanism used in TCP, specially its timeout and exponential back-off mechanism, can lead to self-similarity in aggregate TCP flows.

Queuing analysis focuses on investigating the impact of LRD on the queue length and queuing delay [2, 3, 36, 84, 97, 101, 122]. These research efforts investigate the queuing behavior with long-range dependent input, and indicate the queue behavior will be fundamentally different from that with Markovian input.

Research in the area of traffic control and resource provisioning aims to alleviate the adverse impact of the LRD property. Either closed-loop or open-loop control has been considered. In the open-loop control, the work on queuing analysis with self-similar input has direct bearing on the resource dimensioning problem, and shows the ineffectiveness of allocating buffer space versus bandwidth for self-similar traffic. In closed-loop control (a.k.a. feedback control), the work on multiple time scale congestion control [123, 124] attempts to exploit the correlation structure that exists across multiple time scales in self-similar traffic for congestion control purposes.

Main Theme of the Thesis Most of the aforementioned results regarding the performance impact of the LRD on the Internet seem to indicate that: LRD causes much heavy queuing tails, more packet losses, and longer delays (and hence lower throughput). To some extend, it has been a widely held paradigm that LRD has **adverse** impacts on the Internet. In this thesis, instead of focusing on analyzing the adverse impacts of LRD on the Internet, we take advantage of the scaling property of the Internet traffic and show that the correlation structures that are present in Internet traffic can , if utilized judiciously, be exploited for effective resource and traffic control. The big picture of the thesis is shown in Fig. 1.1. The upper layers schemes modulate the Internet traffic, while the stochastic property of the Internet traffic plays important role in affecting the upper layers' performance. Specifically, we resort to traffic modeling and analysis to understand the stochastic property (called scaling behavior) and utilize the property to facilitate upper layers scheme designs. In what follows, we first elaborate on the adverse impacts caused by LRD, and then outline the research thrusts along which we will take a dramatic turn and effectively exploit the LRD property for resource/traffic control in the thesis.

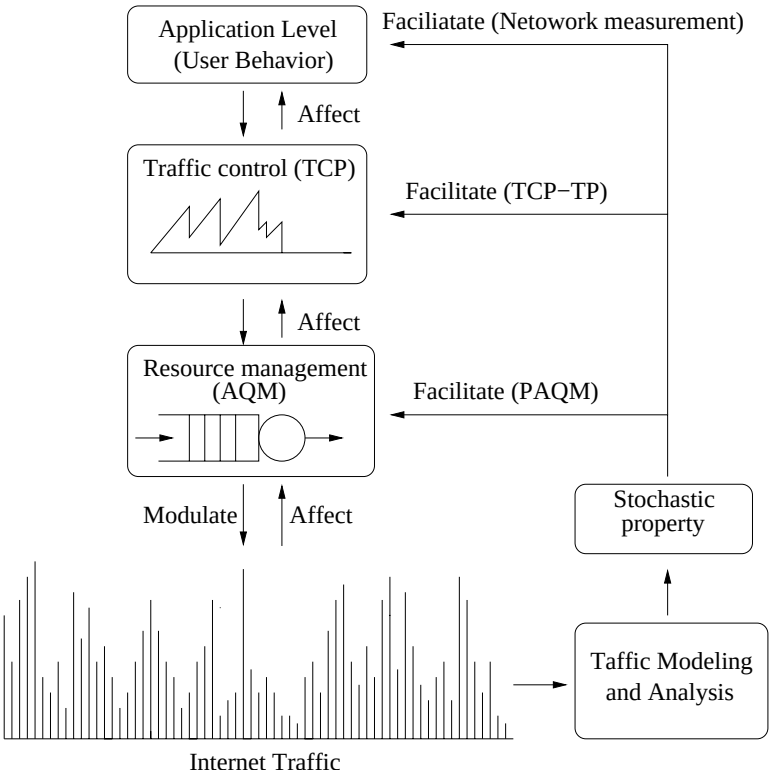


Figure 1.1: The big picture of the main theme of the thesis.

1.1 Adverse Impacts of LRD

As mentioned above, the consequence that Internet traffic exhibits self-similarity is longer queue lengths(with infinite buffer size), and hence larger packet losses and delays in the network. The analytical derivation in [97,121] showed that a queuing system with self-similar traffic as input and a constant service rate would have an infinite queue length. In the case that the value of H is large, a tremendous amount of storage space has to be allocated in order to achieve reasonable increase in the utilization. This also implies that the utilization factor cannot be practically improved by increasing the buffer space.

In addition to the undesirable effect of longer queue sizes, the property of LRD in the Internet traffic also implies the existence of concentrated periods of high activity and low activity (i.e., burstiness) at a wide range of time scales. Scale-invariant burstiness introduces new complexities into resource control and QoS provisioning. On the one hand, burstiness at coarser time scales induces extended periods of either over-utilization or under-utilization. Packets that arrive in the periods of over-utilization experience long delays and are even dropped due to buffer overflow, while packets that arrive in the periods of under-utilization experience the opposite. The large variation in the end-to-end delay packets experience has an adverse impact on transport of QoS-sensitive traffic such as multimedia traffic. On the other hand, if resource reservation is deployed for QoS provisioning, resources have to be reserved with respect to the *peak* rates over a *wide* range of time scales. This adversely affects resource control and degrade the overall performance.

LRD also makes adverse impacts on proactive and passive Internet traffic measurement. In active traffic measurement, probing packets are sent back to back between two end hosts and certain network attributes are inferred by observing the inter-arrival times or the reception statistics of these packets at the receiver. Due to the burstiness of the Internet traffic, an extremely large number of probing packets have to be sent in order to obtain accurate measurements at both small and large time scales. This implies that the probing algorithms have to be be intrusive, i.e., the probing packets themselves may affect the attributes to be inferred(e.g., the available bandwidth of the bottleneck link or the background traffic). In passive traffic measurement such as packet sampling, the inherent heavy-tailedness in LRD makes it difficult to capture the statistics of the Internet traffic.

1.2 Outline of the Thesis

While the LRD of Internet traffic introduces difficulties and complexities into traffic and resource management, it also opens up a new direction for research — the existence of nontrivial correlation structure at larger time scales can be judiciously exploited for better congestion and resource control. How to exploit the abundant correlation structure to improve Internet techniques to achieve better performance is the focus of this thesis. Specifically, we approach the research along the following research thrusts.

First, in order to give a comprehensive explanation of the causes of LRD and multifractality, we present in Chapter 3 a hierarchical model that has an one-on-one correspondence to protocols in the protocol hierarchy of IP networks and sheds insights on how, and to what extent, the user/protocol behavior in each protocol layer contributes to LRD and multifractality. We then prove, with rigorous derivation, that this simple model can explain both LRD at large time scales and multifractality at small time scales. Also, from this model we conclude that the Internet protocol hierarchy makes a significant contribution to the LRD and multifractality properties of Internet traffic. We also analyze the queuing behavior of a fluid queuing system with this hierarchical model as input, and prove that in the long run, multi-fractal traffic has the potential of causing heavier queuing tails than mono-fractal traffic.

Second, we demonstrate in Chapter 4 how the nontrivial correlation structure at larger time scales can be utilized to make prediction based on the history information and how the prediction can be exploited in active queue management (AQM). Specifically, we show that the correlation structure present in long-range dependent traffic can be detected on-line and used to accurately predict the future traffic. We figure in, with the objective of stabling the instantaneous queue length, the prediction results in the calculation of the packet dropping probability. The resulting scheme is termed as *predictive AQM* (PAQM).

To further demonstrate how the information retrieved from the non-trivial correlation structure of Internet traffic can be utilized, we devise in Chapter 5 a novel scheme, called TCP with traffic prediction (TCP-TP), that exploits the prediction result to infer, in the context of AIMD steady-state dynamics, the optimal operational point at which a TCP connection should operate and expedite the process of TCP converging to the optimal operational point. Through analytical

reasoning, we show that the impact of prediction errors on fairness is minimal.

In addition to utilizing the LRD characteristics in AQM and TCP congestion control, we also explore its use in end-to-end measurement in Chapter 6. We present three theoretically sound techniques: prediction, reconstruction and interpolation for proactive measuring the volume of competing cross traffic on an end-to-end path. These methods are designed with the objective of making accurate measurement without introducing an excessively large number of probing packets. we take advantage of not only the correlation structure of Internet traffic, but also its self-similarity (i.e., the same correlation structures are asymptotically present at different time scales). By applying this property we greatly reduce the number of probing packets required to make satisfactory measurement. The proposed methods are implemented and validated both at user level and in the kernel. By comparing results obtained in both implementations, we find that the user level implementation gives slightly over-estimated means of the amount of cross traffic, while the kernel level implementation under-estimates the means, although both implementations give relatively accurate estimates (with the relative error within 10%). We then carry out experiments to evaluate the proposed techniques based on the user level implementation on the Internet. Our experiment results show that all the proposed techniques [1] can capture the mean value of the cross traffic well.

In conjunction with the end-to-end measurement, we perform in Chapter 7 an in-depth, analytical study of three sampling techniques for self-similar Internet traffic, namely static systematic sampling, stratified random sampling and simple random sampling. We show that while all three sampling techniques can accurately capture the Hurst parameter (second order statistics) of Internet traffic, they fail to capture the mean (first order statistics) faithfully, due to the bursty nature of Internet traffic. We also show that static systematic sampling renders the smallest variation of sampling results in different instances of sampling (i.e., it gives sampling results of high fidelity). Based on an important observation, we then devise a new variation of static systematic sampling, called *biased systematic sampling (BSS)*, that gives much more accurate estimates of the mean, while keeping the sampling overhead low. Both the analysis on the three sampling techniques and the evaluation of BSS are performed on synthetic and real Internet traffic traces. The performance evaluation shows that BSS gives a performance improvement of 40% and 20% (in terms of

[1] The interpolation method is slightly modified.

efficiency) as compared to static systematic and simple random sampling.

The contribution of the thesis is attributed in several aspects:

1. The scaling property (self-similarity, LRD, multifractality) of the Internet traffic comes from the infrastructure of the protocol hierarchy of IP networks, where the Internet traffic is generated and modulated. By proposing a hierarchical model, we make it clearer how these complicated phenomena are generated, and to what extent they affect the queuing behavior of the Internet traffic.

2. The nontrivial autocorrelation structure of the Internet traffic can be readily exploited to do accurate prediction. The prediction results can be applied in active queue management, so that the packet drop probability depends not only on the queue length but also on the future incoming traffic obtained by prediction. They can also be utilized to tune the TCP's AIMD phases so as to improve the attainable throughput and reduce packet loss.

3. The scaling property, especially the scale-invariant autocorrelation structure of the Internet traffic shows its power in the domain of network end-to-end measurements. Specifically, the probe packet pattern can be tuned to bring us accurate estimation of the cross traffic on an end-to-end basis while the cost (the number of probe packets) is small.

4. In sampling a self-similar/LRD process with infinite variance, we show that the inherent heavy-tailedness possessed by the LRD traffic helps us to implement a biased systematic sampling (BSS) which amortizes the drawback of traditional sampling methods while keep the important statistics (first and second order moments).

Chapter 2

Background

In this chapter, we summarize the background materials which form the base of the following chapters and introduce the related work. We first give the definition of long-range dependence, self-similarity and heavy-tailed distribution, then we introduce the linear minimum mean square error (*LMMSE*) predictor which is widely used in the following chapters. We validate the *LMMSE* by comparing it with two model-based predictors, i.e., Fractional Brownian Motion (FBM) and Fractal Auto Regressive Integrated Moving Average (FARIMA). We also introduce two simple predictors to amortize the computational heaviness of *LMMSE*. After that, we introduce the notion of multifractality, the renewal processes and renewal density functions. We also outline several relevant known theorems to facilitate the discussion in Chapter 3. At last we summarize the related work.

2.1 Long-Range Dependence and Heavy-tailed Distribution

Let $X = (X_t : t = 0, 1, 2, ...)$ be a covariance stationary (sometimes called wide-sense stationary) stochastic process; that is, a process with constant mean $\mu = E[X_t]$, finite variance $\sigma^2 = E[(X_t - \mu)^2]$[1], and an autocorrelation function $r(k) = E[(X_t - \mu)(X_{t+k} - \mu)], (k = 0, 1, 2, ...)$ that depends only on k. In particular, X is called long-range dependent (LRD) if the autocorrelation function has the form

$$r(k) \sim k^{-\beta} L_1(k), k \to \infty, \tag{2.1}$$

[1]Except for Chapter 7, these two conditions hold throughout the thesis

where $0 < \beta < 1$ and L_1 is slowly varying at infinity, that is, $\lim_{t\to\infty} L_1(tx)/L_1(t) = 1$ for all $x > 0$ (examples of such slowly varying functions are $L_1(t) = const$, $L_1(t) = \log(t)$). For each $m = 1, 2, 3, ...$, let $X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, ...)$ denote a new time series obtained by averaging the original series X over non-overlapping blocks of size m. That is, for each $m = 1, 2, 3, ...$, $X^{(m)}$ is given by

$$X_k^{(m)} = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X_i, k = 1, 2, 3, ... \tag{2.2}$$

Note that for each m, the aggregated time series $X^{(m)}$ defines a covariance stationary process; let $r^{(m)}$ denote the corresponding autocorrelation function.

The process X is called (exactly second-order) self-similar with self-similarity parameter $H = 1 - \frac{\beta}{2}$ if the corresponding aggregated processes $X^{(m)}$ have the same correlation structure as X, i.e.,

$$r^{(m)}(k) = r(k), m = 1, 2, ..., k = 1, 2, ... \tag{2.3}$$

in other words, X is exactly self-similar if the aggregated process $X^{(m)}$ are indistinguishable from X, at least with respect to their second order properties. X is called (asymptotically second-order) self-similar with self-similarity parameter $H = 1 - \beta/2$ if

$$\begin{aligned} r^{(m)}(1) &\to 2^{1-\beta} - 1, m \to \infty \\ r^{(m)}(k) &\to \frac{1}{2}\delta^2(k^{2-\beta}), m \to \infty (k = 2, 3, ...), \end{aligned} \tag{2.4}$$

where $\delta^2(f)$ denotes the second central difference operator applied to a function f, i.e., $\delta^2(f(k)) = f(k+1) - 2f(k) + f(k-1)$. Thus, an asymptotically self-similar process has the property that for large m, the corresponding aggregated time series $X^{(m)}$ have a fixed correlation structure, solely determined by β; moreover, due to the asymptotic equivalence (for large k) of differencing and differentiating, $r^{(m)}$ agrees asymptotically with the correlation structure of X given by Eq. (2.1).

Intuitively, the most striking feature of (exactly or asymptotically) self-similar processes is that their aggregated processes $X^{(m)}$ possess a nondegenerate correlation structure as $m \to \infty$. This behavior is in stark contrast to the more conventional stochastic models, all of which have the

property that their aggregated processes $X^{(m)}$ tend to second order pure noise (as $m \to \infty$), i.e.,

$$r^{(m)}(k) \to 0, m \to \infty, (k = 1, 2, 3...). \tag{2.5}$$

Note that we have chosen the above definitions of self-similarity over the mathematically more convenient definition of a self-similar continuous-time stochastic process $X = (X_t : t \geq 0)$ with stationary increments, namely, for all $a > 0$,

$$X_{at} = a^H X_t \tag{2.6}$$

where equality is understood in the sense of equality of the finite-dimensional distributions, and the exponent H is the self-similarity parameter. Definitions in Eq. (2.4) and Eq. (2.5) have the advantage that they do not obscure the connection with standard time series theory, and they reflect the fact we are mainly interested in large m's (time scales)[2]. Eq. (2.6) will be more useful in analyzing the autocorrelation structure of a self-similar process.

Since LRD is closely related to heavy-tailed distributions, i.e., distributions whose tails decline via a power law with a small exponent (less than 2), we give a succinct summary of heavy-tailed distributions. The most commonly used heavy-tailed distribution is Pareto distribution. A random variable X has Pareto distribution if its pdf follows:

$$f(x) = \alpha k^\alpha x^{-(\alpha+1)}, x \geq k,$$

where α is the shape parameter and determines the decreasing rate of its tail distribution. k is the scale parameter and is the smallest value x can take. The Laplace transform of $f(x)$ is given in [100] (chapter 5) as:

$$L(s) = \int_0^\infty f(x)e^{-sx}dx = 1 - ms + \frac{k^\alpha \Gamma(2-\alpha)}{\alpha - 1}s^\alpha + o(s^\alpha), \tag{2.7}$$

where $m = E(x) = \frac{k\alpha}{\alpha-1}$.

[2]We turn to small time scales in Chapter 3.

2.2 LMMSE Predictor

As we have mentioned in Chapter 1, the non-trivial correlation structure of a LRD process can be leveraged to do prediction, and the predicted results can be used to facilitate resource and traffic control. The proposed predictor is called Linear Minimum Mean Square Error (*LMMSE*) predictor.

Again, let $X(t)$ denote a time series representing the amount of traffic (in bytes) that travels in the network at time point t (seen either by end hosts or by intermediate routers). The predictor keeps track of the aggregate series samples, $X^m(1), X^m(2), \ldots, X^m(n)$, measured in the past n measurement intervals, where $X^m(k)$, $1 \leq k \leq n$, is the aggregate series sample taken in the $(n+1-k)$th most recent interval as defined in Eq. (2.2).

We devise a traffic predictor based on Linear Minimum Mean Square Error (*LMMSE*). Specifically, given the aggregate series $X^m(k), k = 1, ..., n$, where $X^m(k)$ is given in Eq. (2.2), we predict the aggregate series sample, $X^m(n+1)$, in the next interval as a weighted sum of the past n average samples:

$$X^m(n+1) = \begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} X^m(1) \\ X^m(2) \\ ... \\ X^m(n) \end{bmatrix}, \tag{2.8}$$

where $a_1, a_2, ..., a_n$ are the *LMMSE* coefficients and can be expressed as

$$\begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} = \begin{bmatrix} r(n) & r(n-1) & ... & r(1) \end{bmatrix} \begin{bmatrix} r(0) & r(1) & r(2) & ... & r(n-1) \\ r(1) & r(0) & r(1) & ... & r(n-2) \\ ... & ... & ... & ... & ... \\ r(n-1) & r(n-2) & r(n-3) & ... & r(0) \end{bmatrix}^{-1} \tag{2,9}$$

where $r(n)$ is the covariance function of the time series, and can be estimated (due to the property of asymptotically second-order self- similarity) in practice as

$$r(i) = \frac{1}{n} \sum_{t=i+1}^{n} X^m(t) X^m(t-i), 0 \leq i \leq n-1, \tag{2.10}$$

where n is the number of aggregate series samples kept and is a tunable parameter. Note that

the Hurst parameter H that characterizes the LRD property has been implicitly calculated in the covariance function $r(i)$.

The mean square error of the *LMMSE* predictor can be calculated (after a few algebraic operations) as

$$\sigma^2 = \sigma_x^2 - \begin{bmatrix} r(n) & r(n-1) & ... & r(1) \end{bmatrix} \begin{bmatrix} r(0) & r(1) & ... & r(n-1) \\ r(1) & r(0) & ... & r(n-2) \\ ... & ... & ... & ... \\ r(n-1) & r(n-2) & ... & r(0) \end{bmatrix}^{-1} \begin{bmatrix} r(n) \\ r(n-1) \\ ... \\ r(1) \end{bmatrix}, \quad (2.11)$$

where σ_x^2 is the variance of $X(t)$.

Implementation issues To practically implement the *LMMSE* predictor, we consider the following three issues:

1. The *LMMSE* predictor is derived under the assumption of zero mean stationary stochastic process. As the stochastic time series on-line measured may not be of zero mean, we subtract the mean value from the original time series, apply the *LMMSE* predictor to estimate the average of the time series in the next interval, and then add back the mean value.

2. We have to determine how far ahead traffic prediction is made. This translates into the problem of determining an appropriate value, τ, for the interval between two calculation of $X^m(k)$. Under different scenarios, τ should be set to different values, and this will be discussed separately in the following chapters.

3. The operations involved in the calculation of *LMMSE* coefficients (Eqs. (2.9) and (2.10)) are multiplication of time series samples and matrix manipulation, for which fast algorithms exist [1]. Hence they can be readily implemented in hardware. In particular, the author of [1] gave an adaptive algorithm in linear prediction in which the matrix inverse operation in Eq. (2.10) can be avoided. Succinctly, the algorithm starts with an initial estimate of the coefficient a_0. Each time a new data point, $X^m(n)$, is obtained, the algorithm updates a_{n+1} using the recursive equation:

$$a_{n+1} = a_n + \mu \cdot e(n) \cdot X^m(n), \quad (2.12)$$

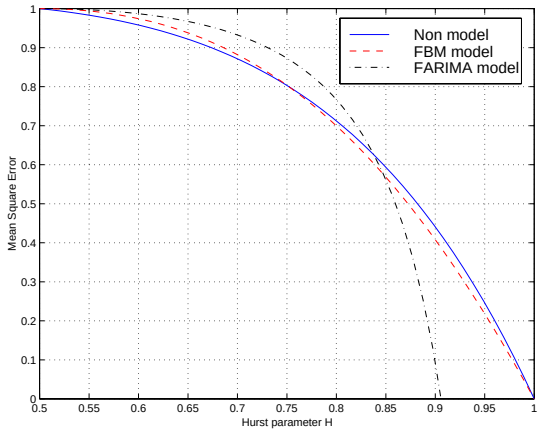


Figure 2.1: Comparison of mean square errors among the *FBM*, *FARIMA*, and *LMMSE* predictors.

where $e(n)$ is the prediction error and μ is a constant. If $X^m(n)$ is stationary, a_i is shown to converge in the mean to the optimal solution [1]. The interested reader is referred to [1] for a detailed account.

Comparison with Fractional Model-Based Predictors Several fractional models and their corresponding predictors have been proposed for time series with LRD characteristics, among which the FBM model and the fractal ARIMA model may have received the most attention. The interested reader is referred to the Appendix A for a brief summary of the two fractional model-based predictors and several important results that are relevant to the discussion of traffic prediction below.

The reason why we use a *LMMSE* predictor in predicting incoming traffic at a router is two fold:

Accuracy: The most important criterion in choosing a predictor is the accuracy. Specifically, let $X(t)$, $\overline{X(t)}$, and $\hat{X(t)}$ represent, respectively, the real traffic, the result of fitting $X(t)$ into a model (*FARIMA* or *FBM*), and the estimated traffic. Then the total predictor error is

$$\begin{aligned} |\hat{X}(t+\tau) - X(t+\tau)| &= |\hat{X}(t+\tau) - \overline{X}(t+\tau) + \overline{X}(t+\tau) - X(t+\tau)| \\ &\leq |\hat{X}(t+\tau) - \overline{X}(t+\tau)| + |\overline{X}(t+\tau) - X(t+\tau)| \\ &= err_{model} + err_{predictor}. \end{aligned} \tag{2.13}$$

That is, the total prediction error consists of the model fitting error, err_{model}, introduced

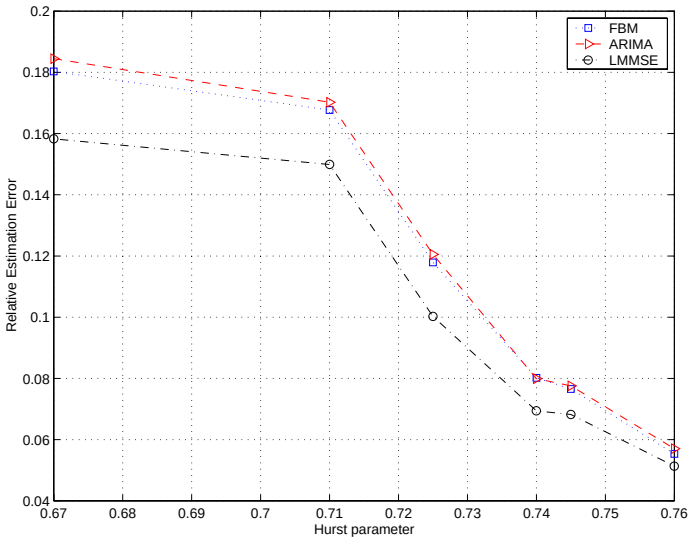


Figure 2.2: Comparison of prediction errors among the *FBM*, *FARIMA*, and *LMMSE* predictors.

by fitting real traffic into a model and the prediction error, $err_{predictor}$, introduced by the predictor itself. The second term can usually be analytically derived, while the first term has to be empirically measured. We depict the second term versus H under the *LMMSE* predictor (Eq. (2.11)/σ_x^2), the FBM predictor (Eq.(A.2)), and the fractal ARIMA predictor (Eq. (A.7)) in Fig. 2.1. When $H \rightarrow 1$ the relative error converges to 0 under all three models, suggesting that the more pronounced the LRD characteristic, the better the performance of predictors. Moreover, the three curves are close to one another when $H \leq 0.85$ (beyond which the curve corresponding to the *FARIMA* model based predictor differs notably). Since analysis of real traffic traces indicates that the H parameter rarely exceeds 0.85 [131], from the theoretic perspective, all three predictors are equally well-suited for Internet traffic prediction.

We have also conducted simulation to evaluate these three predictors in terms of total prediction errors. For illustrative purpose, we depict in Fig. 2.2 the total prediction errors of the three predictors incurred in predicting the composite traffic that traverse a bottleneck link of capacity 20 Mbps and buffer size 100 packets (each of 1000 bytes). Totally 60 connections are established, with each generating packets using the on-off traffic model in *ns-2*. The *LMMSE* makes better prediction than the other two predictors. This is mainly because *LMMSE* is non-model based and hence does not introduce err_{model}.

Ease of implementation: To implement the two fractional model based predictors, one has to on-line estimate the H parameter and engage in complicated calculation of weight coefficients. Specifically, one has to estimate the value of H in the *FBM* model and calculate the weight coefficient in the form of

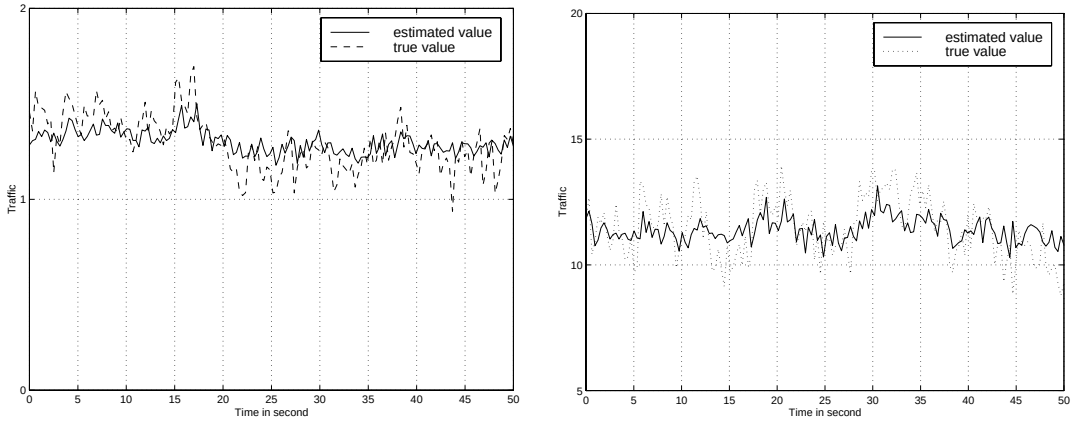
$$\frac{sin(\pi(H-\frac{1}{2}))}{\pi}(-t(T+t))^{-H+\frac{1}{2}}\int_0^a \frac{(\tau(\tau+T))^{H-\frac{1}{2}}}{\tau-t}d\tau,$$

where T is the interval during which the history information is gathered to predict the value at time $T+t$ (Eq. (A.2)). Similarly, one has to estimate the value of $d = H - 1/2$ in the fractal *ARIMA* model and calculate the weight coefficient in the form of

$$\beta_{kj} = -\begin{pmatrix} k \\ j \end{pmatrix}\frac{\Gamma(j-d)\Gamma(k-d-j+1)}{\Gamma(-d)\Gamma(k-d+1)},$$

where $\Gamma()$ is the gamma function (Eq. (A.5)). (The interested reader is referred to the Appendix A for a detailed account.) As a result, it is more difficult to practically implement model-based predictors in router hardware/software. The *LMMSE* predictor, on the other hand, does not require estimation of such parameters, but instead calculates these parameters (in particular, $r(i)$'s in Eq. (2.10)) directly from the collected traffic samples. Moreover, as mentioned in Section 2.2, there exist fast algorithms that can be readily implemented to perform operations involved in the calculation of *LMMSE* coefficients (Eqs. (2.9) and (2.10)) [1].

Validation of the *LMMSE* Predictor To validate the design of the *LMMSE* predictor, we have implemented it in a router in *ns-2* and tested its prediction capability in both single-bottleneck networks and networks of arbitrary topology. We found that the actual traffic and the estimated traffic agree very well. To illustrate this, we depict in Fig. 2.3 the actual traffic over a bottleneck link and its corresponding *LMMSE* estimate. The simulation environment is the same as that in Fig. 2.2 except that each source generates packets using either the on-off traffic model ((a)) or real network traffic traces ((b)). The H parameter estimated under both cases in Fig. 2.3 are 0.75 ((a)) and 0.70 ((b)), respectively. This confirms the LRD characteristic of the traffic. Moreover, under



(a) on-off model (b) real network traffic traces

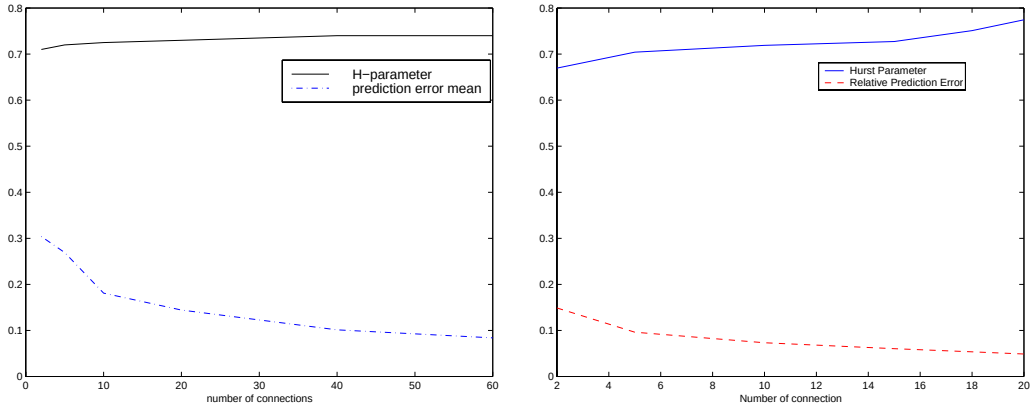
Figure 2.3: Actual and estimated traffic traces when TCP packets are generated using the on-off model or real network traffic traces in *ns-2*.

both cases, the estimated traffic agrees very well with the actual traffic. In the former case, the ratio of the mean estimate error to the actual value is 0.08, while in the latter case, the ratio is approximately 0.15.

To study the effect of the number of connections on the performance, we repeat the above experiment, but vary the number of connections on the bottleneck link. Fig. 2.4 depicts the Hurst parameter and the estimation error versus the number of connections. As shown in Fig. 2.4, the traffic observed on the bottleneck link exhibits the LRD characteristic under both cases, regardless of the number of connections. In particular, the estimation error of the *LMMSE* predictor is less than 0.2 when the number of connections exceeds 10 in the former case, and less than 0.1 when the number of connections exceeds 4 in the latter case.

2.3 The *Simple* Predictor

A major drawback of the *LMMSE* predictor is its computational complexity. As given in Eq. (2.9), calculation of the predictor coefficients requires both matrix inverse and multiplication operations (whose computational computation is $O(n^2)$). Although an iterative algorithm exists (Eq. (2.12)), the coefficients calculated by the algorithm converge only in the mean to their optimal values. To ease the computational burden, we introduce a much simpler predictor, called *simple*. Specifically, similar to the *LMMSE* predictor, we express $\hat{X}^{(m)}(n+1)$ as a linear combination of the past n



(a) on-off model (b) real network traffic traces

Figure 2.4: The Hurst parameter and the estimation error versus the number of connections.

samples, i.e., Eq. (2.8) still holds. (As a result, the discussion on the rationale behind using a non-model-based, *LMMSE*-based predictor vs. model-based predictors in Section 2.2 is also valid here.) However, instead of using Eq. (2.9) to calculate coefficients, we use the following equations

$$\sum_{i=1}^{n} a_i = 1, \tag{2.14}$$

and

$$a_i = (n + 1 - i)^{2H-2}, \tag{2.15}$$

where H is the Hurst parameter.

The rationale behind Eqs. (2.14)–(2.15) is as follows. Let $F(Z)$ denote the Z-transform of $X^m(k)$, and $\hat{X}^m(k)$ the predicted value of $X^m(k)$. Then Eq. (2.8) gives:

$$\hat{F}(Z) = \tilde{H}(Z)\ F(Z), \tag{2.16}$$

where $\tilde{H}(Z)$ is the transfer function of the predictor and has the form:

$$\tilde{H}(Z) = a_1 Z^{-1} + a_2 Z^{-2} + ... + a_n Z^{-n}. \tag{2.17}$$

Eq. (2.14) ensures $\tilde{H}(1) = 1$ (i.e., when the frequency equals 0, the transfer function equals 1) and the predictor is equivalent to a low pass filter. In other words, Eq. (2.14) ensures that the output

	m=1	m=2	m=4	m=6	m=8	m=10	m=12
LMMSE	0.10	0.092	0.080	0.078	0.072	0.075	0.077
Simple	0.096	0.082	0.070	0.067	0.065	0.066	0.068
Trivial	0.12	0.10	0.093	0.084	0.080	0.083	0.086

Table 2.1: Comparison of predictor errors in *LMMSE*, *simple* and *trivial*.

of a direct signal is the signal itself. Eq. (2.15), on the other hand, is based on the fact that for LRD traffic, its autocorrelation function obeys

$$r(\tau) \sim \tilde{H}(2H-1)\tau^{2H-2}. \tag{2.18}$$

Hence we choose the coefficients so that they follow the same law as the autocorrelation function, i.e., we assign heavier weights to history samples that correlate more to the current sample.

There are two parameters, m and n, in *simple* that should be tuned and one parameter, H, that is determined by the traffic and should be on-line measured. To reduce computational complexity, we eliminate the operation of on-line measuring H and set $H = 0.80$. This is based on the observation of our own experiments and several empirical Internet measurements [131] that H is approximately in the range of 0.75 to 0.85 for Internet traffic. On the other hand, to study the impact of setting the parameters m and n on the relative prediction error, we have conducted experiments under a wide variety of network topologies and traffic. In particular, we vary the value of n from 10 to 40, and the value of m from 1 to 12. The first two rows in Table 2.1 gives the total prediction errors under *LMMSE* and *simple* predictors in the same simulation setting as Fig. 2.1, when $n = 10$ and m varies from 1 to 12. (Results under other simulation settings and combinations of parameters give similar trends and hence are not reported here.)

As shown in Table 2.1, *simple* outperforms *LMMSE*, although in theory *LMMSE* is the best linear predictor that gives the minimum mean square error. This is because in reality we have to approximate $r(i)$ as $r^{(m)}(i)$ in Eq. (2.10) in *LMMSE*. This approximation is not accurate especially when i goes to n. As $r(i)$s is heavily used in the calculation of the *LMMSE* coefficients, the performance of *LMMSE* degrades. In contrast, in the *simple* predictor, we take advantage of LRD characteristics and calculate the coefficients using Eqs. (2.14)–(2.15).

The Trivial Predictor As a baseline, we also introduce a *trivial* predictor: $a_1 = 1$ and all the other coefficients are set to zero. In other words, we use $X^m(n-1)$ as the estimate of $X^m(n)$. We conduct the same experiment as in Fig. 2.1 to compare the performance of the *trivial* predictor against that of *LMMSE* and *simple*. The results are given in Table 2.1. As predicted, the performance of *trivial* is the worst among the three predictors. This is because the *trivial* predictor does not make full use of the history information. In Chapter 4 and 5, we will compare the performance of these predictors in the context of PAQM and TCP-TP.

2.4 Multifractality

The multi-scaling property of the Internet traffic can be characterized using *multifractality*. Multifractality refers to the singularity of a signal at different time instances and is usually measured by the *Hölder* exponents. Two types of the *Hölder* exponents have been considered in literatures: one is based on the exponential growth rate and the other on the polynomial approximation. In this chapter, we use the first *Hölder* exponent. The *Hölder* exponent (based on exponential growth rate) of a signal X (X can be a deterministic or random process) at t is defined as:

$$h_X(t) = \liminf_{\epsilon \to 0} \frac{\log \sup_{t_0:|t_0-t|\leq\epsilon} |X(t_0) - X(t)|}{log\epsilon}. \tag{2.19}$$

Conceptually $h_X(t)$ denotes the exponential growth rate of X from the time instant t_0 to the time instant t as t_0 approaches t.

The multifractal spectrum of the signal X for the *Hölder* exponent is defined as:

$$d_X(a) = dim(t > 0 : h_X(t) = a), a \in (0, \infty),$$

where for a set Λ, $dim(\Lambda)$ is the Hausdorff dimension of Λ. Conceptually multifractal spectrum is the measure of points in the interval $(0, \infty)$ at which the *Hölder* exponent takes the value of a. For a mono-fractal process (such as Fractional Brownian Motion), all points take the same value $H = a$, where H is the Hurst parameter of the process, and $d_X(H) = dim(t > 0)$ is the entire interval $(0, \infty)$. If a signal takes different values of *Hölder* exponents at different points in a time

interval, the signal is said to exhibit the multifractal property in this interval.

The multifractal spectrum $d_X(a)$ cannot easily be obtained in reality. Instead, we consider a "histogram" $g(\alpha)$, which measures the number of instants in the traffic that have local *Hölder* exponent α. Furthermore, we use a partition function $\tau(q)$ (the definition of $\tau(q)$ will be given in Eq. (3.49)), to "measure" the extent to which the process exhibits multifractality: if $\tau(q)$ is (close to) linear, the corresponding process follows monofractal scaling; on the other hand, the more concave the shape of $\tau(q)$, the wider the range of local scaling exponents in the traffic. That is, a concave shape of the partition function is consistent with multifractality [47]. In Chapter 3, we propose a novel Internet traffic model which produces traffic possessing the multifractal property.

2.5 Renewal Process and Renewal Function

For a process $X(t)$, let $N_t, t > 0$ be a counting process and Y_n the time between the $(n-1)$st and the nth event of $X(t)$, $n \geq 1$. N_t is a renewal process if the sequence of nonnegative random variables $Y_1, Y_2, ...$ are independent and identically distributed. We define the renewal points to be: $S_0 = 0$, $S_n = \sum_{i=1}^{n} Y_i, n \geq 1$. With all the above definitions, Wald's Equation states $E(S_n) = mE(n)$ where $m = E(Y_1)$.

The renewal function $H(t)$ of a renewal process N_t is defined as $H(t) = E(N_t)$, i.e., the average number of renewals at the time point t. The renewal density function $h(t)$ is defined as

$$h(t) = \frac{dH(t)}{dt} = \lim_{\delta t \to 0} \frac{E(N_{t,t+\delta t})}{\delta t}.$$

Conceptually, $h(t)$ is the mean number of renewals expected in a small interval $[t, t + \delta t]$. For an ordinary renewal process, the Laplace transform of $h(t)$ is given by [31]:

$$h(s) = \frac{F(s)}{1 - F(s)}, \tag{2.20}$$

where $F(s)$ is the Laplace transform of the distribution of Y_1.

The on/off process is a special case of renewal processes, called the alternating renewal process, with the on random variables $Y_1, Y_2, ...$ and the off random variables $Z_1, Z_2, ...$ If we take the ending

point of every off period as a renewal point, then the on/off process has renewal density function $h(t)$ with Laplace transform

$$h(s) = \frac{F_1(s)F_2(s)}{1 - F_1(s)F_2(s)},$$

where $F_1(s)$ and $F_2(s)$ are the Laplace transform of Y_1 and Z_1 respectively.

For the completeness of the thesis, we state three theorems that will be referred to in Chapter 3.

Karamata's Tauberian Theorem

Theorem 1: Let U be a non-decreasing right-continuous real function with $U(x) = 0$ for all $x < 0$. If l varies slowly and $c \geq 0$, $\rho \geq 0$, the following are equivalent:

$$U(x) \sim cx^{\rho} l(x)/\Gamma(1+\rho), x \to \infty, \tag{2.21}$$

$$U(s) \sim cs^{-\rho} l(1/s), s \to 0^{+}, \tag{2.22}$$

where $U(s)$ is the Laplace transform of $U(x)$.

Smith's Theorem

Theorem 2: Suppose $X(t)$ is a renewal process with state space E and A is a measurable subset. For a fixed set A, assume $K(t, A) = \Pr(X(t) \in A, S_1 > t)$ is Riemann integrable. Let $\mu = E(S_1)$, $S_0 = 0$. Then, if $\mu < \infty$,

$$\begin{aligned} \lim_{t\to\infty} P_r(X(t) \in A) &= \mu^{-1} \int_0^{\infty} K(s, A)ds \\ &= \frac{E(\text{occupation time in A in the first cycle})}{E(\text{cycle length})}. \end{aligned} \tag{2.23}$$

Cohen's Theorem

Theorem 3: For a stable GI/G/1 queue with traffic intensity ρ, service time distribution $S(x)$ with mean b, and the content distribution $V(x)$. For $\alpha > 0$, and a slowly varying function $L(x)$,

$$1 - S(x) \sim \alpha b L(x)/x^{\alpha+1} (x \to \infty) \tag{2.24}$$

iff

$$1 - V(x) \sim \rho(1-\rho)^{-1} L(x)/x^{\alpha} (x \to \infty). \tag{2.25}$$

2.6 Related Work

In Chapter 1 we have briefly summarized the research work regarding the scaling property of Internet traffic and summarized the outline of the thesis. In this section we summarize the related work as categorized according to the topics: sources of the scaling property of Internet traffic and Internet traffic modeling, PAQM, TCP-TP and active and passive Internet traffic measurement techniques.

Related Work for the Sources of Scaling Property of Internet Traffic

Since the seminal work of Leland *et al.* [82] laid the groundwork for understanding the self-similarity nature of Internet traffic and its impact on network performance, significant efforts have been made to track down the origin of self-similarity in network traffic. Part of the research along this direction has primarily focused on application level dynamics (e.g., file size) and human factors (e.g., human thinking time) that may contribute to LRD. In particular, Willinger *et al.* [130] indicated the presence of heavy tails in the length of individual flows can be shown to induce LRD in network traffic. Crovella *et al.* [33] cited the distribution of file sizes and the effects of caching and human factors (such as response time and preference) as possible causes for self-similarity in WWW traffic, as file sizes, human thinking time and flow (session) duration all have been shown to exhibit the heavy-tailed property.

Not until recently have protocol and network dynamics been studied as possible causes of LRD in network traffic. Park *et al.* [99] and Feldmann *et al.* [47] pointed out that closed loop protocols like TCP lead to a much richer scaling behavior than open loop protocols like UDP. Veres *et al.* [127] attributed LRD of TCP traffic to the chaotic nature of the TCP congestion control mechanism, and suggested that the adaptive nature of TCP congestion control is one of the causes for LRD on the Internet. Sikdar *et al.* [117] showed how the retransmission and congestion control mechanism in TCP, specifically its timeout and exponential back-off mechanism, can lead to LRD in aggregated TCP flows. Guo *et al.* [59] used a simple Markov chain model to show that when the loss rate is relatively high, the adaptive congestion control mechanism of TCP indeed generates traffic with heavy-tailed OFF (or idle) periods, and therefore introduces LRD into the aggregate traffic.

While LRD or self-similarity (a.k.a. mono-fractal scaling) is characterized by a single scaling

law that holds globally in time and essentially involves only one parameter — the Hurst parameter, multifractal scaling allows for time-dependent scaling laws and hence offers great flexibility in describing irregular phenomena that are local in time. The latter is typically caused by network-specific mechanisms that operate at small time scales and, depending on the state of the network, can have a more or less severe impact on packet dynamics within individual connections. In a first attempt to allow for a more complete description of network traffic, Riedi *et al.* [110] and Ribeiro *et al.* [108] presented traffic models with additive and multiplicative structures. Pursuing in the wavelet domain, they analyze, based on binomial cascades, a multiplicative model that exhibits the multifractal property of network traffic at small time scales and matches the LRD property at large time scales. In essence, cascade models attempt to account for multifractality by viewing networks together with their protocols and controls as a process that fragments units of information in one layer in the networking hierarchy into smaller units in the next layer. These model becomes approximately additive at large scales, as the variance of the cascade generator decreases with the increase in the time scale. This suggests why a purely multiplicative model can be consistent with an additive property in the limit of large scales. Gao *et al.* [56] applied this type of cascade processes to model the measured traffic, and showed that these multifractal processes characterize effectively the long-range dependence properties of the measured traffic. They also consider a queuing system that is fed by such a multifractal process.

Although cascade models are well-suited in explaining multifractality, they lack in the explicit physical meaning in reflecting network protocols (from the application layer to the physical layer) to the corresponding cascade models. As pointed out in [126], the protocol hierarchy may also be a possible cause of multifractality.

Related Work for Traffic Modeling in Characterizing the Scaling Behavior

The studies on the source of the scaling behavior of Internet traffic shed lights on the issue of how to model the Internet traffic. Accordingly, the existing Internet traffic modeling attempts can be grouped into two categories. One category aims to model LRD, while the other targets on multifractality (or both, ours falls in this category).

In the first category, the simple ON/OFF models (or packet trains models) were proposed in [33]

and [131]. It was shown that if the ON (OFF or both) period is heavy-tailed and the number of aggregated flows is large, then the resulted traffic possesses LRD property. As stated in Section 2.6, the heavy-tailedness of the ON (OFF or both) period can be related to the distribution of file sizes or human thinking time, which have been shown to be heavy-tailed.

Fractional Brownian Motion (FBM) model is another well studied one for LRD traffic due to its simplicity (only three parameters involved) and its Gaussian nature facilitates the queuing analysis. The drawback of this model is that it provides a restrictive correlation structure which fails to capture the short-term correlation of real traffic. Chaotic map models [44], [95] explored another way to generate traffic through a deterministic evolution system governed by some specific evolution rules. Although this model can produce LRD traffic with carefully assigned parameters, it is usually difficult to bridge the model and the real traffic parameters. FARIMA models [76] were widely used in video trace modeling. These models can be understood as a filter, with the white Gaussian noise as the input and the output is the LRD traffic we want to generate. These models can capture both the short and the long term correlation of traffic, but it is parameter-sensitive, i.e., it is usually difficult to determine the order and the coefficients of the models (filters).

MMPP (Markov Modulated Poisson Process) [133] models were proposed as an approach to emulate self-similarity over a certain range of time-scales with finite state Markovian models. The basic idea is that the traffic rate is shaped according to the arrival rates in a two state Markov-process; the packet arrives as a Poisson process. In essence, the two state MMPP is a ON/OFF renewal process and the transmission between the two states depends on the distribution of the ON and OFF periods. Although this model is simple, how to fit the model into real traffic parameters is still a non-trivial problem. Similar to MMPP, PPBP(Poisson Pareto burst process) [134] tries to model the Internet traffic using a sequence of bursts (sessions), the bursts arrival process is modeled as a Poisson process, while the duration of each burst follows a heavy-tailed distribution. The problem with the models in the first category is that, all of them can only model the LRD property while lacking the capacity of characterize multifractal properties under small time scales, which triggers the proposals and studies of models in the second category.

In the second category (which is closer to ours), the representatives are MWM (Multifractal Wavelet Model) [110] and cascade [47] models. Both of them are extensions of the FBM mod-

els and use wavelet decomposition to capture the LRD and multifractal properties. As we have mentioned in Section 2.6, these models lack in the explicit physical meaning in reflecting network protocols (from the application layer to the physical layer) to the corresponding cascade models. The model which comes closest in spirit to ours was proposed in [94], called MHOP (Markovian Hierarchical On-off Process). This is a hierarchical model based on the packet transmission process. Although the hierarchical model has clear physical meaning, the Markovian assumption of the on-off component processes is not true and not thoroughly validated. Based on the Markovian on-off processes assumption, the spectral property is easily obtained and shown to be LRD, while the multifractal property was not rigorously proved in the paper. In [87], the authors proposed a new model for multi-scale traffic. They observe that Internet traffic is *scaling* in large time scale which follows Gaussian distribution and multi-scaling in small time scale which follows lognormal distribution. Although this model can capture the scaling (mono-fractal) and multi-scaling (multi-fractal) property of the Internet traffic, the Gaussian and lognormal distribution needs more rigorous demonstration.

For all of the proposed models, none of them possesses clear relationship between the model and the IP network protocol hierarchy and none of them rigorously proved the consistency of LRD and multifractality within the model. Our objective is trying to model the Internet traffic in the most natural fashion and analyzing whether or not this model is equipped with both LRD and multifracatal properties. Our proposed hierarchical model differs from MHOP, MWM and cascade models in:

- no Markovian assumption is made.
- we provide rigorous proof of the LRD and multifractal properties of the traffic generated by the model.
- the model is the first one to have an one-on-one correspondence to protocols in the protocol hierarchy of IP networks.
- the model is validated by real network traces.
- queuing behavior of a queuing system with the traffic generated by the proposed hierarchical model as input is provided.

Related Work for PAQM

AQM refers to the notion of equipping core routers inside a network with the capability to detect incipient congestion and to explicitly signal traffic sources before congestion actually takes place [17]. AQM differs from the traditional drop tail queue in that in a drop-tail queue packets are dropped when the buffer overflows, while in an AQM queue, packets may be dropped before buffer overflow occurs. To illustrate the operations of AQM, we use RED as an example scheme. RED operates by calculating the average queue length, *avg_queue*, upon packet arrival using a low-pass filter with an exponentially weighted moving average. The parameter *avg_queue* is then used as the congestion index. The policy used to detect the likelihood of congestion is characterized by two thresholds, min_{th} and max_{th}. If *avg_queue* is less than min_{th}, the router is considered congestion free and no arrived packets are dropped. On the other hand, if *avg_queue* is greater than max_{th}, all arrived packets are (early) dropped to notify end hosts of (the likelihood of) congestion. When *avg_queue* is between the two thresholds, the probability of (early) dropping a packet linearly increases with *avg_queue* from 0 to p_{max}, where p_{max} is the pre-determined maximum packet dropping probability.

RED was shown in [53] to prevent global synchronization,[3] accommodates bursty traffic, incurs little overheads, and coordinates well with TCP under serious congestion conditions. The performance of RED, however, heavily depends on whether or not the two thresholds are properly tuned. Also, RED was shown to be unfair to individual flows [86], unable to achieve high link utilization and low packet loss ratio simultaneously [7, 51, 77], and exhibit short-term fluctuation in the queue length [98]. In particular, it was shown in [7, 51, 67, 77] that queue length should not be the only parameter to be observed and controlled. Instead, other control variables and policies should be deployed. In Chapter 4, we propose a novel AQM scheme that takes a new angle and manages the queue in anticipation of future incoming traffic.

To alleviate the drawbacks of RED, several AQM schemes have been proposed, e.g., FRED [86], balanced RED (BRED) [6], BLUE [51], stabilized RED (SRED) [98], random exponential marking (REM) [7], PI controller [67], and AVQ [77]. These schemes differ in (1) the performance objectives (in addition to that of notifying end hosts of incipient congestion by dropping/marking packets); (2) the parameters used as an indicator of congestion; and (3) the policies used to detect (incipient)

[3] Global synchronization results from signaling all TCP connections to reduce their congestion windows at the same time, and is usually followed by a sustained period of low link utilization.

congestion and to drop/mark packets. In what follows, we summarize these schemes, and give in Table 2.2 a taxonomy with respect to the above three aspects.

(1) Schemes that aim to achieve per-flow fairness In FRED, a router monitors, not only the global average queue length, but also the average queue length, $qlen_i$, of each individual active connection i. Moreover, two minimum and maximum thresholds are defined for the per-flow average queue length. When a packet from flow i arrives, $qlen_i$ is compared against these two thresholds. A flow with $qlen_i$ less than the minimum limit is not subject to random early dropping even if $minth \leq avg\_queue \leq maxth$. On the other hand, a flow which consistently exceeds the maximum threshold is subject to more aggressive dropping.

BRED extends FRED and imposes three thresholds, ℓ_1, ℓ_2, and ℓ_3, on per-flow queue length, $qlen_i$. The three thresholds divide the space of $qlen_i$ into 4 regions: $(0,\ \ell_1)$, $(\ell_1,\ \ell_2)$, $(\ell_2,\ \ell_3)$, and $(\ell_3,\ \infty)$, each of which is associated with a dropping probability of 0, p_1, $p_2(> p_1)$, and 1, respectively. A router keeps, for each active flow i, the queue length, $qlen_i$, and the number of its packets accepted into the queue since last drop, gap_i. The dropping probability for a packet from flow i is then a function of (i) the region $qlen_i$ is in and (ii) gap_i. The reason for figuring gap_i into the dropping probability is to prevent consecutive multiple drops from a flow. In essence, both FRED and BRED aim to improve the fairness of RED at the expense of keeping per-active-flow state information.

(2) Schemes that decouples the congestion index and the performance index Schemes in this category aim at achieving both high utilization and low packet delay (queue length). The key idea is to decouple the congestion measure from the performance measure. Specifically, these schemes either use additional measures (e.g., link utilization, input rate) as congestion indices, or introduce an intermediate entity (e.g., the price function in REM or the virtual queue in AVQ) so that calculation of the dropping probability is not *directly* related to the actual queue length.

In BLUE, the instantaneous queue length and the link utilization are used as the indices of traffic load, and a single dropping probability p is maintained and used to mark or drop packets upon packet arrival. If the instantaneous queue length exceeds a pre-determined threshold, L, a BLUE router increases p by an amount of *delta* (which is a system parameter). To avoid dropping packets too aggressively, BLUE keeps a minimum interval, *freeze_time*, between two successive

updates of p. Conversely, if the link is idle (i.e., the queue is empty), the BLUE router decreases p by an amount of *delta* periodically (once every *freeze_time*). By adjusting p with respect to the instantaneous queue length and link utilization (idle events), BLUE is shown through simulation to make the instantaneous queue length converge to an operational point with small buffer sizes, while retaining all the desirable features of RED.

REM decouples the congestion measure from the performance measure by defining the price function, $c(k+1)$, as

$$c(k+1) = \max(0, c(k) + \gamma(\alpha(Q(k) - Q_{opt}) + x(k) - R)), \tag{2.26}$$

where $x(k)$ is the aggregate input rate and R is the capacity of the outgoing link. The $(\alpha(Q(k) - Q_{opt})$ term is the queue mismatch, and the $x(k) - R$ term is the rate mismatch. Since $x(k) - R$ measures the rate at which the queue length grows, it can be approximated as $Q(k+1) - Q(k)$, and Eq. (2.26) reduces to $c(k+1) = \max(0, c(k) + \gamma(Q(k+1) - (1-\alpha)Q(k) - \alpha Q_{opt}))$. The price increases if the weighted sum of these mismatches is positive, and decreases otherwise. A REM router calculates the marking probability periodically as $p(k) = 1 - \phi^{-c(k)}$, where ϕ is an arbitrary constant that is greater than 1. Sharing a very similar viewpoint with REM, the PI controller marks each packet with a probability p which is updated periodically using

$$p(k+1) = p(k) + a(Q(k+1) - Q_{opt}) - b(Q(k) - Q_{opt}), \tag{2.27}$$

where $a > 0$ and $b > 0$ are constants chosen according to the design rules given in [67].

The AVQ scheme, on the other hand, takes a dramatically different approach, and uses solely the input rate, $x(t)$, as the congestion index. An AVQ router maintains a virtual queue whose capacity, $\hat{R}$, is adjustable. Upon packet arrival, the virtual queue capacity is updated according to

$$\frac{d\hat{R}}{dt} = \alpha(\gamma R - x(t)). \tag{2.28}$$

where γ is the desired utilization. The rationale behind Eq. (2.28) is to mark/drop packets more aggressively when the arrival rate exceeds the desired utilization (γR) and vice versa. Also, a

fictitious packet is enqueued in the virtual queue if space is available. Otherwise, the fictitious packet is not enqueued and the real packet in the real queue is marked/dropped. The rule for choosing the parameter α is rigorously analyzed using a control theoretic approach to ensure system stability. Through simulation, AVQ is shown to outperform REM and PI in terms of reducing the packet drop rate and average queue length and achieving high utilization.

(3) Schemes that stabilize the instantaneous queue length SRED argues that the instantaneous queue length may fluctuate dramatically under RED if the number of active flows varies. To stabilize the instantaneous queue, SRED equips each queue with a zombie list that keeps a list of M recently seen flows. When a packet arrives, it is compared with a randomly chosen zombie in the zombie list. The result of a hit or a miss is used to detect potential misbehaving flows for more aggressive dropping and to estimate the number of active flows. The estimated value of N is then figured into the calculation of the dropping probability p (e.g., p is an increasing function of N) so as to avoid, upon packet loss, the situation of significant system throughput decrease in the case that there are only a few active flows. The simulation results indicated that SRED keeps the buffer occupancy close to the specified value and away from overflow or underflow.

The work that comes closest to ours is SRED, as both share the same objective of stabilizing the instantaneous queue. Hence, we will make comprehensive performance comparisons between SRED and PAQM in Chapter 4. On the other hand, as a side effect of using the amount of traffic that arrive in the next few intervals to determine the packet dropping probability, PAQM also decouples the congestion measure and the performance measure and can be classified into the second category. Hence, we will also compare PAQM against AVQ (which is reported to give the best performance in the second category) in Chapter 4.

Related Work for TCP-TP (Resource and Traffic Control)

Lots of TCP flavors have been proposed, such as, TCP-Tahoe, TCP-Reno, TCP-NewReno, TCP-Vegas, etc. But the exploitation of LRD in congestion control is still in its infancy. Tuan and Park [123, 124] pioneered the work of exploiting LRD in congestion control, and used, based on the conditional expectation, a simple estimation scheme to explore the correlation structure. The estimate is then used to modulate the magnitude of congestion window increase/decrease in a

heuristic manner in the AIMD algorithm. A related traffic control dimension is *connection duration prediction*. Physical modeling tells us that connections or flows tend to obey a heavy-tailed distribution with respect to their time duration or lifetime, which can be exploited for traffic control purposes. As we know, heavy-tailedness implies that most connections are short-lived, but the majority of traffic is contributed by a few long-lived flows [99]. The idea of employing "connection" duration was first proposed in the context of load balancing in distributed systems where UNIX processes have been observed to possess heavy-tailed lifetimes [60, 61, 80]. The heavy-tailedness renders *predictability*—a connection whose measured time duration exceeds a certain threshold is more likely to persist into the future. In [61] this information was used to do load balancing among the processes. In [116], the discrimination of long-lived flows from short-lived flows was made such that routing table updates can be biased toward long-lived flows and the system stability can be enhanced.

Our TCP-TP scheme follows up Tuan and Park [123, 124]'s work and have taken the heuristic rule-of-thumb to a rigorous plateau, where the traffic prediction is rigorously made with the use of theoretically grounded traffic predictor, and the window adjustment is pinpointed in the context of AIMD steady-state dynamics. Moreover, this is achieved without requiring router support or compromising simplicity and ease of implementation.

Related Work for Network Measurements

Proactive Measurement To facilitate design and development of better resource management protocols, it will be greatly helpful to better understand the dynamic properties and behavior of end-to-end paths in the Internet. Moreover, not to overload routers with traffic measurement and report tasks, it is more desirable for end hosts to infer these properties on an end-to-end basis. Therefore, Sending probing packets to infer the network information (proactive measurement) is a promising candidate to serve the purpose. To this end, several end-host-based or edge-based measurement infrastructure projects (such as IPMA [71], NIMI [90], Felix [49], and Surveyor [118]) and academic research projects (that use the one-packet techniques [74, 115], the packet-pair techniques [14, 22, 105], a combination thereof [79], or the multicast-based inference technique [37, 38, 106]) have been proposed to collect and analyze end-to-end measurements between a number of hosts.

The common feature of the proactive measurement techniques is to inject one or more unicast/multicast *measurement* packets and measure/record the round-trip time (as in the one-packet techniques), the difference in the arrival times of two consecutive packets (as in the packet-pair techniques), or the pattern of packets received in a multicast group (as in the multicast-based inference technique). The measured information is then used to infer the available bandwidth or the packet loss probability, over links of interest. The measurement results can then be utilized for better resource control. For example, the work in [55] [92] enables rate based congestion control based on the estimate of the attainable throughput inferred at the network edge. Cetinkaya *et al.* [23] and Rubenstein *et al.* [112] develop algorithms to perform admission control and detect flows that are subject to the same congestion points. Most of the above approaches, perhaps except [20, 37] (which is grounded on a maximum likelihood estimation approach) and [79] (which is grounded on rigorous algebraic derivation), are devised based on some simple heuristics and observation.

As we have mentioned, the LRD of Internet traffic implies the existence of concentrated periods of high activity and low activity (i.e., burstiness) at a wide range of time scales. A closer investigation also reveals that the existence of *LRD* implies the existence of nontrivial correlation structures at multiple time scales which can actually be exploited to better infer traffic properties. To this end, a multi-fractal-model based cross traffic estimation algorithm, called *Delphi* algorithm, was proposed in [109]. Based on the multi-fractal model, special temporally-spaced probe packets (called "chirp packet trains") were sent and the cross traffic was inferred based on the information thus obtained. The major advantage of the *Delphi* algorithm is that it requires only a small number of probe packets for end-to-end measurement. However, two measurement errors were induced: first, as it is impossible to fit real traffic perfectly into the multi-fractal model without introducing error; and second, the algorithm proposed in [109] to infer the amount of cross traffic is heuristic-based (although with good theoretical reasoning) and also introduces error. The simulation results reported in [109] shown that the performance of the *Delphi* algorithm depends heavily on the bottleneck link utilization.

Based on the understanding of the scaling property of the Internet traffic, in Chapter 6 we proposed three theoretically grounded methods that are either prediction-based or interpolation-based to measure cross traffic of the bottleneck link. In the first method, the future traffic is

predicted based on recent traffic measurements. Only a fixed number of probe packets are sent at the beginning of every T period and LRD-based prediction is made to infer the cross traffic for the remaining time of the period. In the second method, we attempt to reconstruct the entire cross traffic process based on the information obtained by probe packets. Specifically, we use the information to estimate the power spectral density (PSD) of cross traffic, and use inverse Fourier transform to obtain the estimate of the entire process under the assumption that the cross traffic is statistically stationary. In the third method, we periodically send closely-spaced probe packet pairs to "sample" cross traffic of the bottleneck link. Then we interpolate the process between every two sample points. According to the Nyquist criterion, the entire process can be "reconstructed" as long as the sampling rate is at least 2 times as large as the bandwidth of the signal. By virtue of the existence of LRD, the sampling rate can be very low. A finite impulse response (FIR) filter is used to implement the interpolation process under the minimum mean square error criterion.

Passive Measurement—Sampling The major drawback of proactive measurement is the induced overhead of inserting probing packets into the networks, which may have already experienced heavy congestions. Instead of actively sending probing packets, passively sampling the Internet traffic can also provide us valuable information about the Internet conditions.

As has been reported in [39, 48], Internet traffic sampling techniques are very important to understand the traffic characteristics of the Internet. If the sampled results faithfully reflect the real picture of Internet traffic, they can be utilized to monitor traffic on a short-term basis for hot spot and DoS detection [89] or on a long-term basis for traffic engineering [48] and accounting [40]. As such, the packet sampling approaches have been suggested by the IETF working groups IPFIX [70] and PSAMP [102]. Tools such as NetFlow [28] employ a naive 1-out-of-N sampling strategy in the router design.

The major challenge in employing sampling techniques is, however, scalability. Inspecting each individual packet for each flow or sampling at a very high rate is obviously not feasible, due to the large volume of traffic. On the other hand, if the sampling rate is not adequate, the sampled results may not reveal actual traffic characteristics. What makes the problem even more difficult is the bursty nature (LRD) of the Internet traffic. In the context of packet sampling, the burstiness of Internet traffic implies that either the sampling rate must be high enough or the sampling strategy

has to be judiciously devised so as to capture all the peaks and valleys in the traffic. As oversampling increases the memory requirements for the off-board measurement devices, and has the danger of making the sampling method unscalable, the latter approach (devising a sampling strategy that is able to capture the traffic characteristics) is preferred.

Several research efforts have been made to investigate the effectiveness of sampling techniques in measuring network traffic. Three commonly used sampling techniques, i.e., static systematic[4], stratified random and simple random, have been studied by Claffy *et al.* [26]. In particular, they explored various time-driven and event-driven sampling approaches with both random and deterministic selection patterns at a variety of time granularities. The results showed that event-driven techniques outperform time-driven ones, while the differences within each class are small. Cozzani and Giordano [32] used the simple random sampling technique to evaluate the ATM end-to-end delay. Estan and Varghese [46] proposed a random sampling algorithm to identify large flows, in which the sampling probability is determined according to the inspected packet size. Duffield *et al.* [40] focus on the issue of reducing the bandwidth needed for transmitting traffic measurements to a back-office system for later analysis and devise a size-dependent flow sampling method. The notion of adjusting the sampling density upon detection of traffic changes in order to meet certain constraints on the estimation accuracy was proposed in [25]. Finally, Duffield *et al.* [41, 42] investigated the issue of inferring stochastic properties of original flows (specifically the mean flow length, and the flow length distribution) from the sampled flow statistics.

In spite of all the research efforts, none has taken into account of the self-similarity of Internet traffic in devising sampling strategies. Three of the most important parameters for a self-similar process are the mean (first order statistics), the Hurst parameter (second order statistics), and the average variance of the sampling results. In particular, the average variance of the sampling results is defined as follows: let $\bar{X}$ be the real mean of the parameter of interest in the original process, and X_i be the sampled result in the ith instance of sampling (i.e., the ith experiment). Then the average variance is defined as $E(V) = E[E[(X_i - \bar{X})^2]]$, where the inner expectation is taken over all the samples in one instance of sampling, and the outer expectation is taken over all the sampling instances (e.g., different starting sampling points in the systematic sampling technique

[4]In Chapter 6, we omit "static" and simply name it systematic.

give different sampling instances). The mean gives the most direct value of the traffic attribute to be measured. The Hurst parameter characterizes the second order statistics for a self-similar/LRD process, and is crucial for queuing analysis. The average variance is an index of the fidelity of the sampling results.

Although it has been reported in [100] that in sampling self-similar process with the three commonly used sampling techniques, the sampled mean is always smaller than the actual mean (i.e., the sampling techniques under-estimate the mean), no solution has been proposed to address this problem. The issues of whether the various sampling techniques accurately capture the Hurst parameter and/or render a small average variance have not been studied either. In this thesis (Chapter 7) we seek to close the gap and

T1. Investigate whether or not the three commonly used sampling techniques accurately capture the Hurst parameter. We also provide a sufficient and necessary condition (*SNC*) that a sampling strategy must satisfy in order to maintain the autocorrelation structure of the original process. Our derivation indicates that all the three methods satisfy the *SNC*.

T2. Verify whether or not the three commonly used sampling techniques render small average variances (and hence give high fidelity) by leveraging the results reported in [29]. Our research finding is that the systematic sampling method outperforms the other two.

T3. Demonstrate all three methods cannot provide accurate estimate of the mean for self-similar Internet traffic, especially when the sampling rate is small. We then propose, based on an important observation, a new variation of the systematic sampling technique, called *biased systematic sampling* (BSS), that gives much more accurate estimates of the mean, while keeping the sampling overhead low. As BSS is a variation of the systematic sampling technique, it retains all the advantages of the latter.

Category	Scheme	Congestion index	Policies used to detect congestion and to drop/mark packets
Achieving fairness	FRED [86]	queue length	Monitors the per-flow queue length and fine-tunes the dropping/marking decision w.r.t. the per-flow queue length.
	BRED [6]	queue length	Defines three thresholds and divides the state of per-flow queue length into 4 regions. Fine-tunes the dropping/marking decision w.r.t. the per-flow state.
Achieving high link utilization and low queue length	BLUE [51]	queue length, link idle event	Increases p if the instantaneous queue length exceeds L and has not been updated for over *freeze_time*. Decreases p if the link is idle for over *freeze_time*.
	REM [7]	queue length, input rate	Defines the price function, $c(k)$, as in Eq. (2.26), and calculates the marking probability as $p(k) = 1 - \phi^{-c(k)}$.
	PI [67]	queue length, input rate	Calculates the marking probability as in Eq. (2.27).
	AVQ [77]	input rate	Maintains a virtual queue. At each packet arrival, enqueue a fictitious packet and update the virtual queue capacity using Eq. (2.28). Mark/drop a real packet only if the virtual queue overflows.
Stabilizing queue	SRED [98]	queue length	Keeps a zombie list to keep track of recently seen flows, to detect misbehaving flows, and to estimate the number, N, of active flows. Figures in N in the packet dropping probability.
	PAQM	queue length, amount of future traffic	Exploits the long range dependency characteristics to predict the future enqueuing rate, $\hat{f}(k+1)$. Figure in $\hat{f}(k+1)$ in the packet dropping probability with the objective of stabilizing the queue.

Table 2.2: A taxonomy of AQM schemes.

Chapter 3

A Hierarchical Model for Internet Traffic

In this chapter, we aim to give a comprehensive explanation of the scaling behavior (self-similarity/LRD and multifractality) of the Internet traffic. Specifically, we propose a simple hierarchical model that has an one-on-one correspondence to protocols in the protocol hierarchy of IP networks. We envision the traffic governed by protocols as hierarchical on/off processes. At the up-most level, the hierarchical model is an ON/OFF model with an ON period corresponding to the active period induced by, for example, a HTTP request and an OFF period to the user's thinking time. Each ON period is, in turn, composed of one or more level-1 on and off periods, with an level-1 on period corresponding to a TCP connection for transmitting embedded object(s) and an off period to the time incurred in three-way handshaking and other processing delays. Each level-1 on period is, in turn, composed of TCP slow start, congestion avoidance, and timeout/exponential back-off phases and hence can be further decomposed into level-2 on and off periods. In essence, an on period that corresponds to the active period in a protocol layer can be decomposed into smaller on and off periods to account for activities carried out in the lower protocol layer.

We validate the hierarchical model with traces gathered at IRCache.net [72] and Lucent Technologies Bell Labs [88]. In particular, we identify all the ON/OFF periods and level-1/level-2 on/off periods within the traces and fit them into heavy tailed distributions. We prove via analytic derivation and empirical studies that this simple model does exhibit LRD and multifractality. Finally, we analyze the queuing behavior of a fluid queuing system with this hierarchical model as input and prove that in the long run, multifractal traffic has the potential of causing heavier queuing tails

Notation	Definition of Notations
B, B_i	ON period
I, I_i	OFF period
m_{ON}, m_{OFF}	mean value of ON and OFF period
N, N_i	number of level-1 on and off period in an ON period
b, b_i	level-1 on period
i, i_i	level-1 off period
G, H_I	CDF of B and I
M, P, Q	CDF of N,b, and i
γ_t, a_t	input rate process and input process of the hierarchical model
$r(t)$	autocorrelation function of γ_t
$D(t)$	variance of a_t
$S_k(q), \tau(q)$	structure function and partition function of the traffic generated by the hierarchical model
S_n	renewal sequence of a ON/OFF renewal process
$C(t), T_s$	content process and empty time of a single server queue system
$U(x), V(x)$	CDF of $C(S_n)$ and $C(t)$

Table 3.1: Notations used throughout the chapter.

than mono-fractal traffic. To the best of our knowledge, this is the first work that links protocol behaviors in the protocol stack with ON/OFF models in a natural way and thus provides a vehicle to mathematically studying the causes of LRD and multifractality across the protocol stack.

The rest of the chapter is organized as follows. In Section 3.1 we present and validate our hierarchical model. Following that, we prove in Sections 3.2- 3.3 several important properties of the model in explaining LRD and multifractality properties of network traffic. In Section 3.4 we apply the model as an input to a fluid single sever queue (SSQ) and derive the asymptotic queuing behavior. Finally, we conclude the chapter in Section 3.5.

3.1 Proposed Hierarchical Model

Before delving into the description and analysis of our hierarchical model, we give in Table 3.1 a list of notations used throughout the chapter.

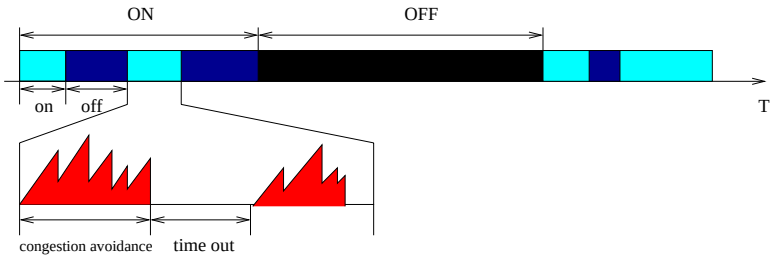


Figure 3.1: The hierarchy model.

3.1.1 Single on/off Source Model

The ON/OFF model has been widely used to model sources that intermittently generate/transmit signals. In the ON/OFF model, a source can be in one of the two states: ON and OFF. An ON period, B_i, $i \geq 0$ corresponds to the period in which the source is active transmitting, while an OFF period, $I_i, i \geq 0$, corresponds to the period in which the source is idle. The process alternates between B_i and I_i, $i = 0, 1, 2, \ldots$ and is an alternating renewal process.

We assume that the random vectors $(B_i, I_i), i \geq 0$ are independent and identically distributed. We also assume that B_i and I_i are independent. Because of the i.i.d. nature of (B_i, I_i), we may omit in the subsequent discussion the subscripts as long as this does not give ambiguity. Let G and H_I denote, respectively, the CDF of B and I.

3.1.2 Hierarchical Model

The hierarchical model is an ON/OFF model. Each ON period defines an active period in which a protocol carries out its activities. The ON period consists of several smaller, level-1 on and off periods, each of which accounts for the protocol activities carried out in the next protocol layer. Each level-1 on period, in turn, consists of several even smaller, level-2 on and off periods that account for activities carried out in the next protocol layer.

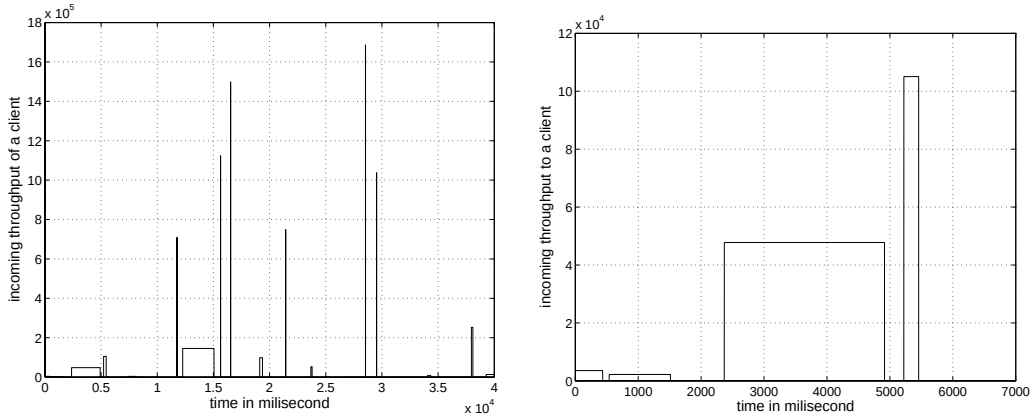
We use HTTP to illustrate the model. Starting from the application layer, the time instant at which a user issues an HTTP request corresponds to the starting point of an ON period (Fig. 3.1). In response to a HTTP request, one or more webpage(s) will be sent back to the requester. Each webpage contains several objects. If HTTP 1.0 is used, one TCP connection is established for each object. Otherwise if HTTP 1.1 is used, all the objects in a webpage will be transmitted via a single, pipeline TCP connection. In both cases, each ON period consists of one or more smaller, level-1

on and off periods, with an on period corresponding to a TCP connection that transmits one or more embedded objects, and an off period to the time incurred in three-way handshaking and other processing delays. Each level-1 on period is, in turn, composed of slow start, congestion avoidance, and occasionally timeout and exponential back-off phases. That is, each level-1 on period can be further decomposed into level-2 on and off periods. The decomposition can proceed as desired. For example, if we take into account of the contention and back-off activities carried out in the MAC layer protocols, each level-2 on period can be further decomposed according to the specific MAC protocols. We define the *level* of the hierarchical model to be the number of times an ON period is decomposed, and label the up-most ON/OFF model as level-0.

For tractability of analysis, we consider a two-tier hierarchical model, i.e., the ON period is divided in to level-1 on and off periods[1]. However, the analytical results can be easily extended into higher-tier hierarchical models. The two-tier model is characterized by the following random variables: (i) the ON period B, (ii) the OFF period I, (iii) the number of level-1 on and off periods in the i^{th} ON period, denoted as N_i, (iv) level-1 on periods denoted as $b_n, 0 \leq n \leq N_i$, and (v) level-1 off periods denoted as $i_n, 0 \leq n \leq N_i$. Again, we assume b_n and i_n are i.i.d and hence the index n can be omitted. We also assume N_i's are i.i.d., i.e., the numbers of embedded objects in different ON periods follow the same distribution and are independent.

Let M, P and Q denote the CDF of N, b, and i, respectively. The remaining question is what distribution G, H_I, M, P, and Q should follow respectively. Since B is determined by N, b and i, we only need to determine the distributions for H_I, M, P and Q. Barford *et al.* [8] conducted a comprehensive analysis of real Internet traces and model matching, and indicated that all the four random variables follow heavy-tailed (Pareto or Weibull) distributions with specific shaping and scale parameters. Furthermore, if a higher-tier model is used, the heavy-tailed distribution can also be used to model level-2 on and off periods. This is because an level-1 on period that comprises TCP slow start and congestion avoidance phases and an level-1 off period caused by the TCP timeout in a TCP connection are also shown in [117] and [59] to exhibit the heavy-tailed behavior. Similarly, the back-off mechanism (such as CSMA/CD) used in the MAC layer also causes level-3 off periods to exhibit the heavy-tailed behavior in a level-2 on period. Hence, we assume in this

[1]In Sections 3.2 and 3.4 we focus on the two-tier hierarchical model, while in Section 3.3 we consider a k-tier hierarchical model, where k can goes to infinity.



(a) Incoming traffic

(b) incoming traffic at smaller time scale

Figure 3.2: The incoming traffic at a Web client.

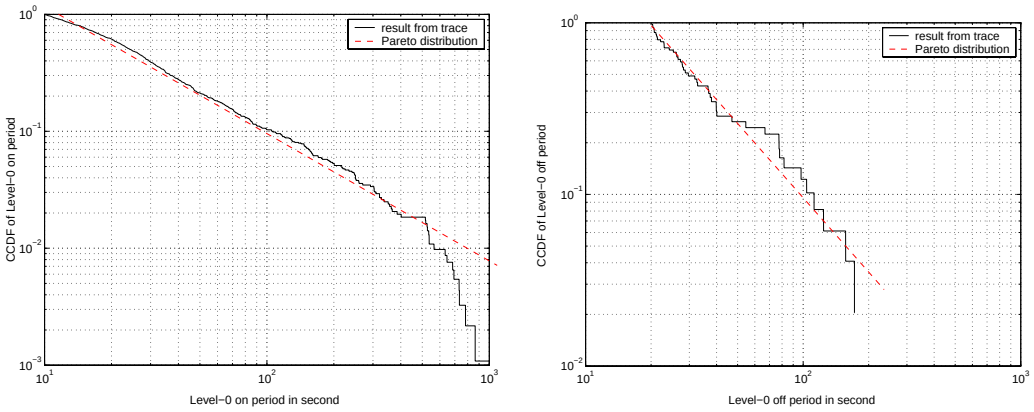
chapter H_I, M, P and Q follow the Pareto distribution with different scale and shape parameters.

3.1.3 Validation of the Model

In this section, we aim to validate our model in two steps. First, we use the traces from IRcache to validate level-0 ON/OFF periods. Then, we use the packet level traces from Lucent Technologies Bell Labs to validate level-1 and level-2 on/off periods.

Validation of the Level-0 ON/OFF Model

As discussed above, several newly measured Web traffic traces were used in [8] to determine the statistical property (CDF and PDF) of the ON and OFF periods. This is done by fitting the traces to some known distributions, such as Pareto or Weibull distributions. We apply the similar methodology and use the Web workload traffic traces gathered from *IRCache.net* [72] to validate the model. The set of traces contains all the traffic (HTTP requests and Web data) transfered between Web servers and Web clients. For each client, the trace provides the following information: starting time of the transmission of a web object (HTML, jpg, etc), the time elapse for that transmission, and the size of the web object. We use the traces collected by nine servers on January 22, 2003, and extract the incoming traffic at an individual client. Fig. 3.2 gives one representative of incoming traffic at a client.



(a) Level-0 ON,$\alpha = 1.1, k = 10$ (b) Level-0 OFF,$\alpha = 1.9, k = 20$

Figure 3.3: Model fitting for level-0 ON and OFF periods.

In Fig. 3.2 (a), we show that the incoming traffic in the first 40 second period of the trace. Each spark/bar corresponds to the transmission of a web object (i.e. a TCP connection). We use the time interval t between two objects (sparks/bars) to identify a level-0 ON period: whenever $t < T_c$, we say the two objects are within one HTTP transmission, and hence within the same level-0 ON period; otherwise, the two objects are considered to reside in two different pages, and the interval between their transmissions is considered as a level-0 OFF period. The value of T_c is set to be 10 seconds in our validation study, because Barford *et al.* have shown in [8] that the time interval between two objects in downloading a webpage is typically less than 10 seconds. Fig. 3.2 (b) gives an enlarged view of the same traffic trace in $[0, 7]$ second. The level-0 ON period approximately around 5 second consists of four smaller level-1 on/off periods. To fit the traces into the hierarchical model, we characterize these periods with the Pareto distributions. Fig. 3.3 gives the model fitting result for ON and OFF periods, with the perspective parameters given in the figure.

Validation of Level-1 and Level-2 on/off Periods

As the IRcache traces only provide information at the TCP flow level, they cannot be used to validate our model at higher levels. Instead, we use the traces obtained by Lucent Technologies Bell Labs [88] on March 8, 2000. The traces are in the *tcpdump* format, and contain detailed packet level information for hundreds of pairs of end hosts. We identify 765 web clients (with anonymous IP addresses) from which thousands of TCP connections were initiated. The initiator of a TCP

connection is identified and used to track down all the packets in the TCP connection. Fig. 3.4 gives the sample trace of one such TCP connection extracted from the trace. We scan all the TCP connections initiated by a web client and assume they correspond to the same HTTP request.[2] We then extract: (1) level-1 on periods that correspond to the durations of TCP connections; (2) level-1 off periods that correspond to intervals between two connections; (3) level-2 on periods that correspond to the TCP slow start and congestion avoidance phases; (4) level-2 off periods that correspond to TCP timeouts intervals. Level-1 on and off periods can be readily identified as the *tcpdump* traces provide such information. To identify level-2 on and off periods, we calculate the time interval t between two events (indexed by P,S or F in the trace, Fig. 3.4). If $t < 500ms$ we say the two events are in the same level-2 on period; otherwise we say a timeout occurs and the interval is taken as a level-2 off period. The reason why the threshold of 500 ms is used is due to the fact that the TCP timeout value is usually set between 0.5 to 1 second.

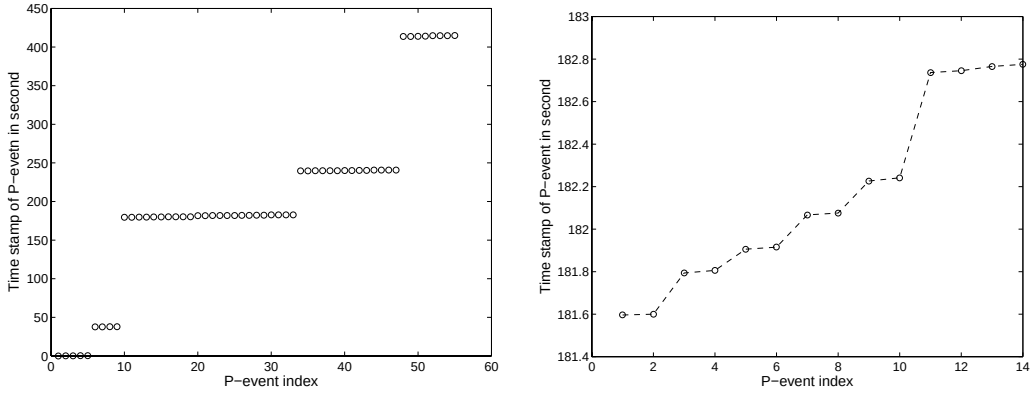
Fig. 3.5 (a) gives TCP connections at a web client. Each small circle represents an event and each segment corresponds to a TCP connection. The difference in the y-axis between each two segments represents an level-1 off period (as the y-axis gives the time stamp when an event occurs) and the difference in the y-axis between the first event and the last event in a segment represents a level-1 on period. Fig. 3.5 (b) gives an enlarged version of the third segment in Fig. 3.5 (a). Using the $500ms$ threshold, we identify that there is at least one timeout (i.e., level-2 off period) in this TCP connection (the jump between the 10th and 11th events). We extract all the level-1 and level-2 on/off periods for all the 765 web clients and fit these periods with Pareto distributions. Figs. 3.6–3.7 give the model fitting results.

We also extracted the number of TCP connections initiated by a web client and correspond this to the number, N, of level-1 on periods in a level-0 ON period. We fit its distribution to a Pareto distribution with $\alpha_N = 1.8$ and $K_N = 2.3$. The result is shown in Fig. 3.8.

[2]This assumption may not be true in reality, and hence in our analysis we discard level-1 off periods that are extremely large (greater than 30s).

```
06:00:04.686121 0.0.0.12.402: S 1390028000:1390028000(0)
06:00:04.716714 0.0.0.12.402: P 1:50(49)
06:00:04.812549 0.0.0.12.402: P 50:64(14)
06:00:04.824797 0.0.0.12.402: P 64:79(15)
06:00:04.917533 0.0.0.12.402: P 79:87(8)
06:00:04.926380 0.0.0.12.402: P 87:107(20)
06:00:05.114469 0.0.0.12.402: P 107:132(25)
06:00:05.160996 0.0.0.12.402: P 132:155(23)
06:04:04.429655 0.0.0.12.402: P 156:181(25)
06:04:04.444582 0.0.0.12.402: P 181:202(21)
06:04:04.939855 0.0.0.12.402: P 202:228(26)
06:04:04.948373 0.0.0.12.402: P 228:247(19)
06:04:04.968683 0.0.0.12.402: P 247:253(6)
06:04:04.978518 0.0.0.12.402: F 253:253(0)
```

Figure 3.4: A sample trace of a TCP connection. The first column is the timestamps of events, the second column is the IP address of the client, the third column is the type of events (S: SYN, P: received packets, F: FIN), and the last column is the sequence number of packets sent or received.

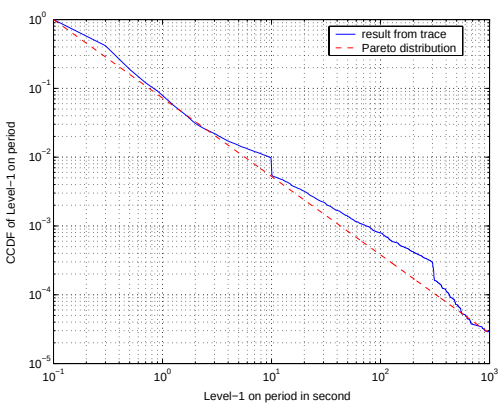


(a) all TCP connections (b) one TCP connection

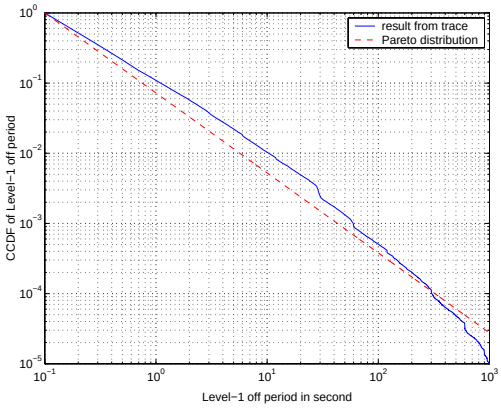
Figure 3.5: TCP connections at a Web client.

3.2 Long Range Dependence Property of the Hierarchical Model

In the previous section we validate our model using real Internet traces. In this section we investigate whether or not such a hierarchical model is consistent with LRD at large time scales and multifractality at small time scales. In other words, we intend to verify whether the traffic generated by this model exhibits LRD and multifractality simultaneously. As has been studied in [110] and [108], a model with additive and multiplicative structures has such attributes. In general, LRD and multifractality do not contradict with each other, as LRD is related to the second order statistic property of a wide sense stationary (WSS) process, while multifractality is concerned with the local singular behavior of a process which may or may not be a WSS process. An additive model

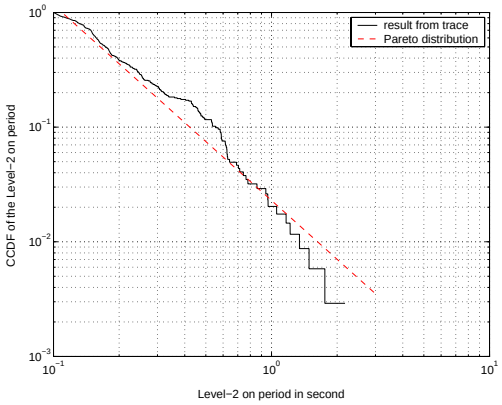


(a) Level-1 on $\alpha = 1.14, k = 0.1$

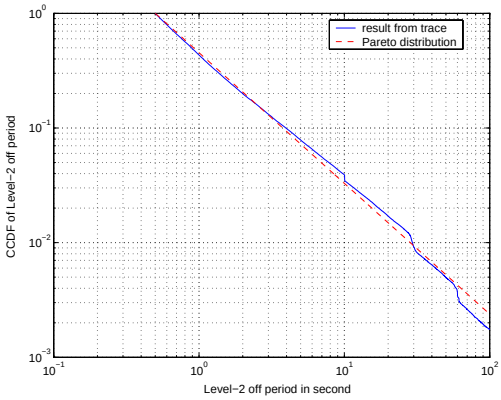


(b) Level-1 off $\alpha = 1.25, k = 0.1$

Figure 3.6: Model fitting for level-1 on and off periods.



(a) Level-2 on $\alpha = 1.78, k = 0.1$



(b) Level-2 off $\alpha = 1.15, k = 0.5$

Figure 3.7: Model fitting for level-2 on and off periods.

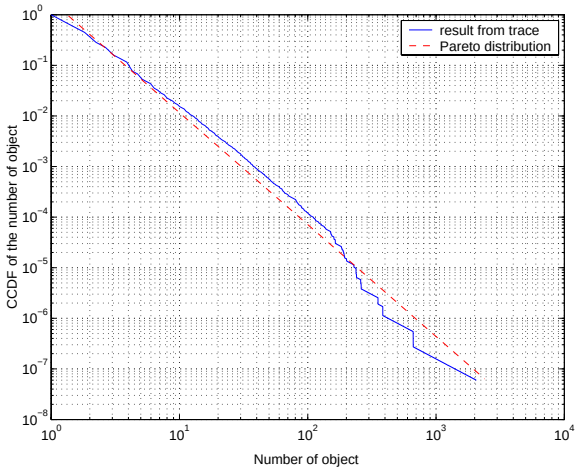


Figure 3.8: The number of objects is fit into a Pareto distribution with $\alpha_N = 1.8$.

such as an ON/OFF renewal process can be strictly stationary if the initial ON/OFF distributions are carefully chosen, and LRD is simply a by-product of the heavytailedness of the ON/OFF periods. On the other hand, a multiplicative model such as the cascade model can generate extremely complicated local behaviors when the time scale is arbitrarily small and multifractality is a natural result. As long as a multiplicative model becomes additive at large time scales, such a model can hold both the LRD and multifractality properties. Our hierarchical model is obviously multiplicative at small time scales and tends to be additive at large time scales (at level 0, the process is simply a simple ON/OFF alternating renewal process), and hence we postulate that it possesses the two properties. We prove in this section our hierarchical model possesses the LRD property, and defer the proof of its multifractality property to Section 3.3.

In order to analyze the LRD property of the hierarchical model, we need to derive (i) the distribution, G, of an ON period B and (ii) the autocorrelation function of the process generated under the hierarchical model.

3.2.1 Distribution of G

By the definition given in Section 3.1 we have

$$B = (b + i) \times N, \tag{3.1}$$

where b, i and N have Pareto distributions with shape parameters α_b, α_i and α_N respectively and scale parameters K_b, K_i, and K_N respectively. Let the Laplace transform of B, I, N, b, and i be denoted as $G(s)$, $H_I(s)$, $M(s)$, $P(s)$ and $Q(s)$ respectively. Then we have

$$\begin{aligned} G(s) = E(e^{-Bs}) &= \int_0^\infty E(e^{-Bs}|N = x)d(M(x)) \\ &= \int_0^\infty (P(s)Q(s))^x d(M(x)). \end{aligned} \tag{3.2}$$

Let $m_b \triangleq E(b)$, $m_N \triangleq E(N)$ and $m_i \triangleq E(i)$. By applying Eq. (2.7), and let $A(s) \triangleq P(s) \cdot Q(s)$. Then $G(s)$ can be further expressed as

$$\begin{aligned} G(s) &= \int_0^\infty A(s)^x d(M(x)) = \int_0^\infty e^{x \log(A(s))} d(M(x)) = M(-\log(A(s))) \\ &= M(-(\log(P(s)) + \log(Q(s)))). \end{aligned} \tag{3.3}$$

Let $m_b \triangleq E(b)$, $m_N \triangleq E(N)$ and $m_i \triangleq E(i)$. Then given that both P and Q are Pareto distributions and $1 \leq \alpha_b \leq 2$, $1 \leq \alpha_N \leq 2$, and $1 \leq \alpha_i \leq 2$, we have (by Eq. (2.7))

$$P(s) = 1 - m_b s + \frac{K_b^{\alpha_b} \cdot \Gamma(2-\alpha_b)}{\alpha_b - 1} s^{\alpha_b} + o(s^{\alpha_b}) \tag{3.4}$$

$$M(s) = 1 - m_N s + \frac{K_N^{\alpha_N} \cdot \Gamma(2-\alpha_N)}{\alpha_N - 1} s^{\alpha_N} + o(s^{\alpha_N}) \tag{3.5}$$

$$Q(s) = 1 - m_i s + \frac{K_i^{\alpha_i} \cdot \Gamma(2-\alpha_i)}{\alpha_i - 1} s^{\alpha_i} + o(s^{\alpha_i}) \tag{3.6}$$

Combining Eqs. (3.3), (3.4), (3.5) and (3.6), we obtain:

$$G(s) = 1 + m_N \cdot \Xi + \frac{K_N^{\alpha_N} \cdot \Gamma(2-\alpha_N)}{\alpha_N - 1} \cdot (-1)^{\alpha_N} \cdot \Xi^{\alpha_N} + o(\Xi^{\alpha_N}), \tag{3.7}$$

where $\Xi = Y + Z$ with

$$Y = \log(1 - m_b s + \frac{K_b^{\alpha_b} \cdot \Gamma(2-\alpha_b)}{\alpha_b - 1} s^{\alpha_b} + o(s^{\alpha_b})), \tag{3.8}$$

and

$$Z = \log(1 - m_i s + \frac{K_i^{\alpha_i} \cdot \Gamma(2-\alpha_i)}{\alpha_i - 1} s^{\alpha_i} + o(s^{\alpha_i})). \tag{3.9}$$

Note that as $x \to 0$, $\log(1+x) = x - \frac{x^2}{2} + o(x^2)$. As a result Y and Z in Eqs. (3.8)–(3.9) can be expressed as:

$$Y = -m_b s + \frac{K_b^{\alpha_b} \Gamma(2-\alpha_b)}{\alpha_b - 1} \cdot s^{\alpha_b} - \frac{1}{2} \cdot m_b s^2 + o(s^2), \tag{3.10}$$

and

$$Z = -m_i s + \frac{K_i^{\alpha_i} \Gamma(2-\alpha_i)}{\alpha_i - 1} \cdot s^{\alpha_i} - \frac{1}{2} \cdot m_i s^2 + o(s^2). \tag{3.11}$$

Then,

$$\begin{aligned}\Xi &= Y + Z \\ &= -(m_b + m_i)s + \frac{K_b^{\alpha_b}\Gamma(2-\alpha_b)}{\alpha_b - 1} \cdot s^{\alpha_b} \\ &\quad + \frac{K_i^{\alpha_i}\Gamma(2-\alpha_i)}{\alpha_i - 1} \cdot s^{\alpha_i} - \frac{1}{2}(m_b + m_i) \cdot s^2 + o(s^2). \end{aligned} \tag{3.12}$$

By Eq. (3.7) and the fact that $1 \leq \alpha_i \leq 2$, $1 \leq \alpha_b \leq 2$, and $1 \leq \alpha_N \leq 2$, we have

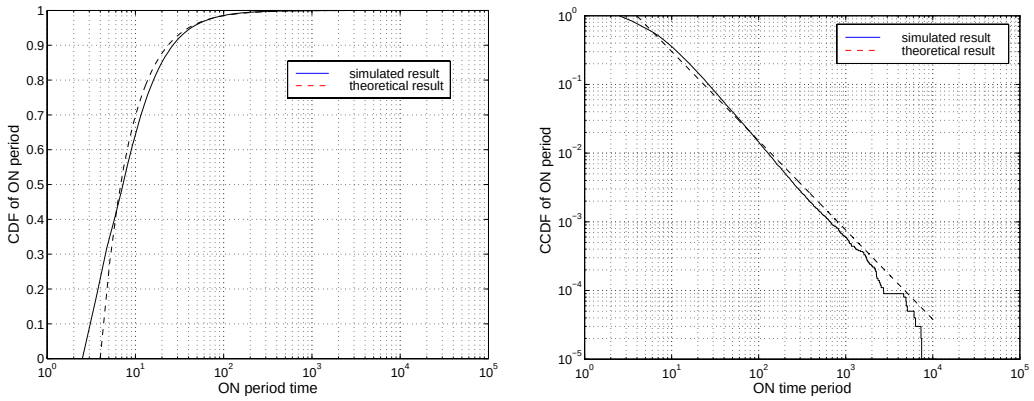
$$\begin{aligned} G(s) &= 1 - m_N(m_b + m_i)s + m_N \frac{K_b^{\alpha_b}\Gamma(2-\alpha_b)}{\alpha_b - 1} \cdot s^{\alpha_b} \\ &\quad + m_N \frac{K_i^{\alpha_i}\Gamma(2-\alpha_i)}{\alpha_i - 1} \cdot s^{\alpha_i} + (m_b + m_i)^{m_N} \cdot \frac{K_N^{\alpha_N}\Gamma(2-\alpha_N)}{\alpha_N - 1} \cdot s^{\alpha_N} + o(s^2) \\ &= 1 - m_N(m_b + m_i) + C \cdot \frac{\Gamma(2-\alpha)}{\alpha - 1} \cdot s^{\alpha} + o(s^{\alpha}), \end{aligned} \tag{3.13}$$

where $\alpha = min(\alpha_b, \alpha_i, \alpha_N)$ and $C = C(\alpha_N, \alpha_b, \alpha_i, K_N, K_b, K_i)$ is a constant determined by α_N, α_b, α_i, K_N, K_b, and K_i. Eq. (3.13) states that B can also be characterized as a random variable with the Pareto distribution with the shape parameter being the minimum of the shape parameters of b, N and i.

In order to validate Eq. (3.13), we carry out the following experiments using Matlab. We generate level-1 on and off periods according to the Pareto distributions M, P, and Q with $\alpha_N = 1.4, K_N = 2$, $\alpha_b = 1.2, K_b = 1.5$, and $\alpha_i = 1.5, K_i = 1.5$, and obtain the CDF of the resulting ON period B. We also generate the CDF of a Pareto distribution with $\alpha = min(\alpha_N, \alpha_b, \alpha_i) = 1.2, K = 1$. As shown in Fig. 3.9, both CDF's (and their corresponding complementary CDF's) match very well.

3.2.2 Autocorrelation Function of the Model

Although the second order statistics of an alternating ON/OFF process has been thoroughly studied in lots of literatures, the one of a hierarchical alternating ON/OFF process has never been seen in publications. In this section, we make efforts to give a rigorous derivation of the autocorrelation structure of such a two-tier hierarchical alternating ON/OFF process.



(a) CDF (b) CCDF

Figure 3.9: CDFs and CCDFs of ON-periods obtained by simulation and by theoretical results.

As shown in Section 3.2.1, the ON period in the two-tier hierarchical model still follows Pareto distribution.

Let $\{\gamma_t\}_{t\geq 0}$ be the stationary process representing the input rate generated under the two-tier hierarchical model at time t, and

$$a_t = \int_0^t \gamma_s ds \tag{3.14}$$

be the input process generated during $(0, t)$. We further denote $\{\Gamma_t^N\}$ and $\{A_t^N\}$ as the sum of N i.i.d. copies of the processes $\{\gamma_t\}$ and $\{a_t\}$, i.e., $\Gamma_t^N = \sum_{i=1}^N \gamma_t^i$, and $A_t^N = \sum_{i=1}^N a_t^i$.

For a fixed t, γ_t is in the ON state with probability μ, and in the OFF state with probability $1 - \mu$. In an ON period, γ_t is in the level-1 on state with probability μ_h and in the level-1 off state with probability $1 - \mu_h$. Hence, it is straightforward to obtain that $E(\gamma_t) = \mu \cdot \mu_h$ and $\sigma^2 = Var(\gamma_t) = \mu\mu_h(1 - \mu\mu_h)$, i.e., γ_t is a Bernoulli random variable with mean $\mu\mu_h$ and variance $\mu\mu_h(1 - \mu\mu_h)$. The autocorrelation function of γ_t can be expressed as:

$$r(t) = E((\gamma_0 - \mu\mu_h)(\gamma_t - \mu\mu_h)) = E(\gamma_0\gamma_t) - (\mu\mu_h)^2. \tag{3.15}$$

Since γ_t is a Bernoulli random variable, we have:

$$E(\gamma_0\gamma_t) = P_r(\gamma_0 = 1 \text{ and } \gamma_t = 1) \stackrel{\triangle}{=} \Pi_{11}(t). \tag{3.16}$$

a_t, on the other hand, is a random variable taking values in $(0, t)$ and with mean $\mu\mu_h t$ and variance:

$$D(t) \stackrel{\triangle}{=} Var(a_t) = Var(\int_0^t \gamma_u du) = 2\int_0^t \int_0^v r(u)dudv. \tag{3.17}$$

What is left is the derivation of $\Pi_{11}(t)$ (Eq. (3.16)). Let the CCDF of B be denoted as $\overline{G}(x) = 1 - G(x)$ and the CCDF of b as $\overline{P}(x) = 1 - P(x)$. Recall that the ON/OFF process is a renewal process with the renewal density function $h_{ON}(t)$. We consider a stationary version of the renewal process [107], in which the initial delay interval, $\tilde{B}$, of the ON period is the forward recurrence time and has the distribution:

$$\Pr(\tilde{B} > x) = \int_x^\infty \frac{\Pr(B > s)}{m_B} ds. \tag{3.18}$$

Let the CCDF of $\tilde{B}$ be denoted as $\overline{\tilde{G}}(x) = 1 - \tilde{G}(x)$. In each ON period, the level-1 on and off periods also constitute a (truncated) renewal process with the renewal density function $h_{on}(t)$. Similarly, we consider a stationary version of the renewal process in which the initial delay interval, $\tilde{b}$, of the level-1 on period is the forward recurrence time, and has the distribution:

$$\Pr(\tilde{b} > x) = \int_x^\infty \frac{\Pr(b > s)}{m_b} ds. \tag{3.19}$$

Let the CCDF of $\tilde{b}$ be denoted as $\overline{\tilde{P}}(x) = 1 - \tilde{P}(x)$.

Under the above notations, we have

$$\begin{aligned}\Pi_{11}(t) = \Pr(\tilde{b} > t) &+ \int_0^t h_{on}(u)\overline{P}(t-u)du \cdot \Pr(\tilde{B} > t) \\ &+ \int_0^t h_{ON}(u) \cdot \Big(\overline{\tilde{P}}(t-u) \\ &+ \int_u^t h_{on}(v)\overline{P}(t-v)dv \cdot P_r(B > t-u)\Big) du,\end{aligned} \tag{3.20}$$

where the three terms respectively account for the probabilities that

1. the initial on period is greater than t.
2. given the initial ON period is greater than t, one level-1 off period ends in $(u, u + \delta u)$, for some $u < t$, and the on period that follows has a period with the length larger than $t - u$.

3. given the initial ON period is less than t, one level-0 OFF period ends in $(u, u+\delta u)$, for some $u < t$, and the ON period that follows has a period with the length larger than $t-u$. During this ON period, t can fall in the first level-1 on period, or in a level-1 on period after some level-1 off period, so it consists of two items that have similar meanings as the first two items.

Let A_3 represent the third term in Eq. (3.20) and $S(t) \stackrel{\triangle}{=} h_{on}(t) \star \overline{P}(t)$, where $\star$ denotes the convolution operation. Then, A_3 can be re-written as

$$\begin{aligned} A_3 &= \int_0^t h_{ON}(u)\overline{\tilde{P}}(t-u)du + \int_0^t h_{ON}(u)\overline{G}(t-u) \cdot \int_u^t h_{on}(v)\overline{P}(t-v)dvdu \\ &= h_{ON}(t) \star \overline{\tilde{P}}(t) + \int_0^t h_{ON}(u)\overline{G}(t-u) \cdot (h_{on}(t-u) \star \overline{P}(t-u))du \\ &= h_{ON}(t) \star \overline{\tilde{P}}(t) + \int_0^t h_{ON}(u) \cdot (\overline{G}(t-u)S(t-u))du \\ &= h_{ON}(t) \star \overline{\tilde{P}}(t) + h_{ON}(t) \star (\overline{G}(t) \cdot S(t)). \end{aligned} \tag{3.21}$$

Hence, we get:

$$A_3 = h_{ON}(t) \star \overline{\tilde{P}}(t) + h_{ON}(t) \star (\overline{G}(t) \cdot S(t)). \tag{3.22}$$

Finally we have

$$\Pi_{11}(t) = \overline{\tilde{P}}(t) + (h_{on}(t) \star \overline{P}(t)) \cdot \overline{\tilde{G}}(t) + h_{ON}(t) \star \overline{\tilde{P}}(t) + h_{ON}(t) \star (\overline{G}(t)S(t)). \tag{3.23}$$

Let $\Pi_{11}(s)$ be the Laplace transform of $\Pi_{11}(t)$. By renewal theory [31] it is straightforward to obtain

$$\begin{aligned} \Pi_{11}(s) &= \frac{m_b s - 1 + P(s)}{m_b s^2} + \frac{h_{on}(s)(1-P(s)) \star (m_B s - 1 + G(s))}{m_B s^3} \\ &\quad + \frac{h_{ON}(s)(m_b s - 1 + P(s))}{m_b s^2} + h_{ON}(s)\Phi(s), \end{aligned} \tag{3.24}$$

where

$$\Phi(s) = \overline{G}(s) \star S(s) = \frac{1-G(s)}{s} \star (\frac{h_{on}(s)(1-P(s))}{s}). \tag{3.25}$$

$P(s)$ and $G(s)$ are Laplace transform of b and B respectively, and by the results summarized in

Section. 2.5, $h_{on}(s)$ and $h_{ON}(s)$ can be expressed respectively as:

$$h_{on}(s) = \frac{Q(s)(1-P(s))}{m_b s(1-P(s)Q(s))}, \tag{3.26}$$

and

$$h_{ON}(s) = \frac{H_I(s)(1-G(s))}{m_B s(1-G(s)H_I(s))}. \tag{3.27}$$

Let I_1, I_2, I_3 and I_4 denote the four terms in Eq. (3.24). To analyze the asymptotic behavior of $r(t)$ as $t \to \infty$, we let $s \to 0$. To facilitate the discussion, we introduce the following Lemma.

Lemma 1: Given $\alpha > -1$, $\beta > -1$, we have:

$$\lim_{t\to 0} \int_0^t u^\alpha (t-u)^\beta du \sim t^{\alpha+\beta+1} \tag{3.28}$$

where $a(t) \sim b(t)$ means $\lim_{t\to 0} \frac{a(t)}{b(t)} = C$, and C is a constant. *Proof:* Let $Y(t) = \int_0^t u^\alpha (t-u)^\beta du$. Then $Y(s) = \frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{s^{(\alpha+\beta+2)}}$ and $Y(t) \sim Ct^{\alpha+\beta+1}$ (because $\alpha+\beta+1 > -1$), where $C = \frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)}$.

To simplify I_1 to I_4, we define $\omega_x = \frac{K_x^{\alpha_x}\Gamma(2-\alpha_x)}{\alpha_x - 1}$, where x can be B, b, I or i. By Eq. (2.7) and by plugging Eqs. (3.26)-(3.27) in the four terms of Eq. (3.24), it is straightforward to show We then rewrite $h_{ON}(s)$ and $h_{on}(s)$, after some simple calculation as:

$$\begin{aligned} h_{on}(s) &= \frac{m_b - \omega_b s^{\alpha_b - 1} - m_b m_i s + m_i \omega_b s^{\alpha_b} + \omega_i m_b s^{\alpha_i}}{m_b(m_b+m_i)s} + o(s^{\alpha_i \wedge \alpha_b - 1}) \\ &= \frac{\alpha(s)}{m_b(m_b+m_i)s} + o(s^{\alpha_i \wedge \alpha_b - 1}), \end{aligned} \tag{3.29}$$

with $\alpha(s) = m_b - \omega_b s^{\alpha_b - 1} - m_b m_i s + m_i \omega_b s^{\alpha_b} + \omega_i m_b s^{\alpha_i}$, and

$$\begin{aligned} h_{ON}(s) &= \frac{m_B - \omega_B s^{\alpha_B - 1} - m_B m_I s + m_I \omega_B s^{\alpha_B} + \omega_I m_B s^{\alpha_I}}{m_B(m_B+m_I)s} + o(s^{\alpha_I \wedge \alpha_B - 1}) \\ &= \frac{\beta(s)}{m_B(m_B+m_I)s} + o(s^{\alpha_I \wedge \alpha_B - 1}), \end{aligned} \tag{3.30}$$

with $\beta(s) = m_B - \omega_B s^{\alpha_B - 1} - m_B m_I s + m_I \omega_B s^{\alpha_B} + \omega_I m_B s^{\alpha_I}$. The Laplace transform of G, H_I,

P, and Q can then be expressed as:

$$G(s) = 1 - m_B s + \omega_B s^{\alpha_B} + o(s^{\alpha_B}), \tag{3.31}$$

$$H_I(s) = 1 - m_I s + \omega_I s^{\alpha_I} + o(s^{\alpha_I}), \tag{3.32}$$

$$P(s) = 1 - m_b s + \omega_b s^{\alpha_b} + o(s^{\alpha_b}), \tag{3.33}$$

$$Q(s) = 1 - m_i s + \omega_i s^{\alpha_i} + o(s^{\alpha_i}). \tag{3.34}$$

Plugging Eqs. (3.29), (3.30) and (3.31)–(3.34) in the expressions of I_1 through I_4 and by applying Lemma 1 we have

$$I_1 \sim \frac{\omega_b}{m_b} s^{\alpha_b - 2}, \tag{3.35}$$

$$\begin{aligned} I_2 &= \frac{\alpha(s)(m_b - \omega_b s^{\alpha_b - 1} + o(\alpha_b - 1))}{m_b(m_b + m_i)s} \star (\frac{\omega_B}{m_B} s^{\alpha_B - 2} + o(s^{\alpha_B - 2})) \\ &\sim \frac{\omega_B m_b}{m_B(m_b + m_i)} s^{\alpha_B - 2}, \end{aligned} \tag{3.36}$$

$$\begin{aligned} I_3 &= \frac{\beta(s) \cdot \omega_b}{m_B m_b (m_B + m_I)} \cdot s^{\alpha_b - 3} + o(s^{\alpha_b - 3}) \\ &\sim \frac{m_I \omega_b}{m_B m_b (m_B + m_I)} \cdot s^{\alpha_b - 2}, \end{aligned} \tag{3.37}$$

$$I_4 = \frac{\beta(s)}{m_B(m_B + m_I)s} \cdot I_2 \sim \frac{\omega_B m_b}{(m_b + m_i)(m_B + m_I)} \cdot s^{\alpha_B - 2}. \tag{3.38}$$

Plugging Eqs. (3.35)-(3.38) into Eq. (3.23), we have

$$\begin{aligned} &\Pi_{11}(s) = I_1 + I_2 + I_3 + I_4 \\ &\sim \frac{\omega_b}{m_b} s^{\alpha_b - 2} + \frac{\omega_B m_b}{m_B(m_b + m_i)} s^{\alpha_B - 2} \\ &\quad + \frac{m_I \omega_b}{m_B m_b (m_B + m_I)} \cdot s^{\alpha_b - 2} + \frac{\omega_B m_b}{(m_b + m_i)(m_B + m_I)} \cdot s^{\alpha_B - 2} \\ &\sim C_\alpha s^{\alpha_{min} - 2}, \text{as } s \to 0, \end{aligned} \tag{3.39}$$

where $\alpha_{min} = min(\alpha_b, \alpha_B)$ and $C_\alpha = \frac{\omega_b}{m_b} + \frac{m_I \omega_b}{m_B m_b (m_B + m_I)}$ if $\alpha_{min} = \alpha_b$; $= \frac{\omega_B m_b}{(m_b + m_i)(m_B + m_I)}$ otherwise.

By applying Karamata's Tauberian Theorem (Section 2.5) and the fact that $\alpha_B = \min(\alpha_b, \alpha_i, \alpha_N)$,

we have:

$$r(t) \sim ct^{2-\alpha_B} L(t), \text{as } t \to \infty, \tag{3.40}$$

where $L(t)$ is a slowly varying function.

By Eq. (3.17) the Laplace transform of $D(t)$ is $D(s) = 2\frac{r(s)}{s^2}$ and hence

$$D(s) \sim cs^{\alpha_B - 4} L(\frac{1}{s}). \tag{3.41}$$

By Eq. (3.41) and by applying the extended form of Karamata's Tauberian Theorem [12] [119], we know that the Tauberian condition holds, i.e.,

$$\lim_{\lambda \downarrow 1} \lim \inf_{t \to \infty} \inf_{u \in [1,\lambda]} \frac{D(ut) - D(t)}{t^{3-\alpha_B} L(t)} \geq 0 \tag{3.42}$$

Therefore, we have

$$D(t) \sim ct^{3-\alpha_B} L(t), \text{as } t \to \infty. \tag{3.43}$$

By Eqs. (3.40) and (3.43) we conclude that as $t \to \infty$, i.e., at large time scales, the model exhibits LRD with $H = \frac{3-\alpha_B}{2}$. Since the aggregate traffic consists of i.i.d. individual sources with the autocorrelation function given in Eq. (3.40), its autocorrelation function has the same type of autocorrelation structure as in Eq. (3.40), i.e., at large time scales, the traffic is LRD with $H = \frac{3-\alpha_B}{2}$. We validate the above conjecture by *ns-2* simulation. We generate aggregate traffic that is composed of n on/off sources, where n varies from 10 to 1000. Each individual source is governed by the hierarchical model with α_B changing from 1.1 to 1.9. We calculate the Hurst parameter of the aggregate traffic using a wavelet based tool provided by Abry *et al.* [111] and compare it against the theoretical result $H = \frac{3-\alpha_B}{2}$. For different values of n, we obtain similar results and hence only show the results in the case that $n = 100$. As shown in Fig. 3.10, both the simulation and theoretical results agree well, i.e., the proposed hierarchical model can generate traffic that exhibits the LRD property at large time scales. One important observation is that the index (Hurst parameter) of the LRD traffic is determined by the heaviest (smallest shape parameter α) component contained in this two-tier hierarchical model. Our conjecture for N-tier ($N \geq 3$) hierarchical model is that the Hurst parameter of the LRD traffic generated by the model

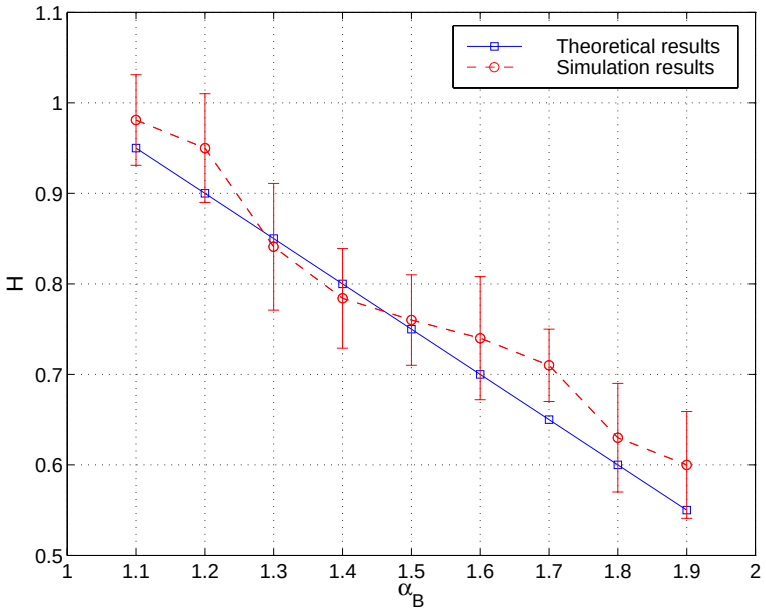


Figure 3.10: An example that shows the proposed hierarchical model generates traffic that exhibits the LRD property at large time scales.

is determined by the heaviest component in the model.

3.3 Multifractal Property of the Hierarchical Model

Recall that if a function has different *Hölder* exponents at different time points in an interval, it has the multifractal property (see Section 2.4). In this section, we show that the proposed hierarchical model generates traffic with the multifractal property. This is done by deriving the multifractal spectrum of the traffic generated by the model. Specifically, we first show that the multifractal spectrum of the aggregate traffic is the same as that of an individual connection. Then we derive the multifractal spectrum of an individual connection by showing that our model is consistent with multiplicative cascade and deriving a "histogram" $g(\alpha)$ and a partition function $\tau(q)$. Recall that $g(\alpha)$ measures the number of instants in the traffic that have local *Hölder* exponent α, while $\tau(q)$, gives the extent to which the process exhibits multifractality (see Section 2.4).

We consider an infinite number of sources that generate traffic in compliance with the proposed hierarchical model. We define T_k to be the transmission starting time of source k, and assume T_k is a sequence strictly increasing to ∞. Each source generates traffic of amount J_k. For clarity

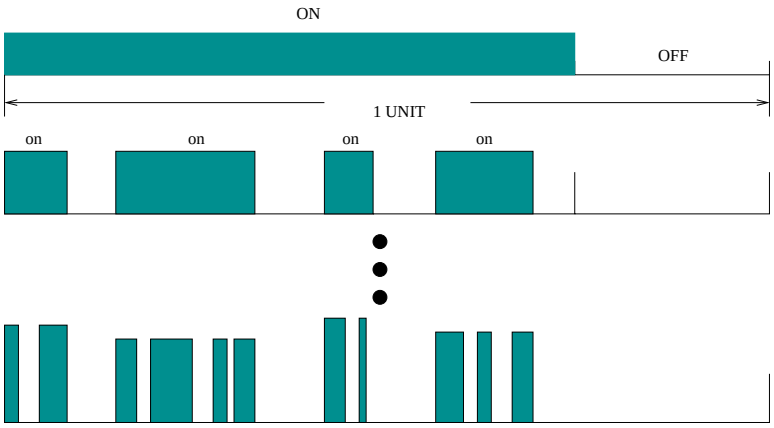


Figure 3.11: The cascade of the traffic in one time unit.

of notation, we assume the amount of traffic J_k has been translated to the total duration of on periods. We also define a_t^k to be the cumulative amount of data transmitted by source k in $[T_k, t]$. a_t^k is a non-decreasing cadlag function (left continuous with right hand limits) starting at 0 and increasing to ∞. The aggregate cumulative traffic generated by all sources in $[0, t]$ can then be expressed as

$$X(t) = \sum_{k=1}^{k=\infty} a_{t-T_k}^k. \tag{3.44}$$

Let d_X be the multifractal spectrum of $X(t)$ and d_{a_1} the multifractal spectrum of a_t^1. As discussed in [93], if a_t^k's are identically distributed with stationary increments and the multifractal spectrum of a_t^k is not degenerated to a single point and is non-random for any non-random interval, then $d_X = d_{a_1}$. That is, the aggregate traffic $X(t)$ inherits the multifractal structure of individual sources. This implies that (i) if multifractality exists in the aggregate traffic, it is caused by intermittency of individual transmissions, presumably caused by on and off periods in the hierarchical model; and (ii) we only need to focus on one such source and calculate its multifractal spectrum.

Usually a construction that fragments a given set into smaller and smaller pieces according to some geometric rule (e.g., binomial decomposition) and, at the same time, divides the measure (e.g., the amount of traffic) of these pieces according to some other (deterministic or random) rule is called a multiplicative process or cascade. This type of multiplicative process or cascade may result in processes with multifractality. In what follows, we show the proposed hierarchical model is also a type of multiplicative cascade (Fig. 3.11).

We follow the following sequence to derive the multifractal spectrum of the hierarchical model and show its multifractal property: (i) we derive the length of an on period at level k; (ii) we derive an approximate of the *Hölder* exponent (defined in Eq. (2.19)); (iii) we define the multifractal spectrum $g(\alpha)$ and related it to the partition function $\tau(q)$; and (iv) we demonstrate the multifractal property of the traffic generated by the hierarchical model by showing the concavity of $\tau(q)$.

Average length of an on period at level k We consider a n-level hierarchical model. Since multifractality is concerned with the irregular behavior of a process at small time scales, we focus on one time interval and normalize it to be one time unit. Let M_0^0 denote the random variable that represents the measure (i.e., the sending rate in our model) in the ON period at level 0 and has the mean M_0, W_a^i be the average number of level-i on/off periods in a level-$(i-1)$ on period($W_a^0 = 1$), and $M_{i,j}^k$ be the measure in the i^{th} on period of the j^{th} level-$(k-1)$ on period at the k^{th} level, where $0 \leq j \leq W_a^{k-1} - 1$ and $0 \leq i \leq W_a^k - 1$. Suppose at level 0, one unit of traffic is transmitted. Since in the OFF period the source is idle, the one unit of traffic is send during the ON period, with the measure M_0^0 (with mean $M_0 = \frac{m_{ON}+m_{OFF}}{m_{ON}}$, where m_{ON} and m_{OFF} are the mean values of ON and OFF respectively). The traffic in the ON period is subsequently decomposed into level-i on/off periods, $i \geq 1$. Let $M_{i,j}^k$ be a random variable with the mean $M_k = \prod_{i=0}^{k-1} M_i$, and $M_i = \frac{m_{on}^i + m_{off}^i}{m_{on}^i}$ (except $i = 0$), where m_{on}^i and m_{off}^i are the mean values of the on and off periods at level i.

Then the average interval of an on period at level k is

$$\zeta_k = \tilde{u}^{-1} \prod_{i=0}^{k} J_i = \hat{u}^{-k} \prod_{i=0}^{k} J_i, \tag{3.45}$$

where, $\tilde{u} = \prod_{i=0}^{k} W_a^i \stackrel{\triangle}{=} \hat{u}^k$, $J_0 = \frac{m_{ON}}{m_{ON}+m_{OFF}}$, and $J_i = \frac{m_{on}^i}{m_{on}^i + m_{off}^i}$ for $i > 0$. By assuming that all J_i's are equal to the same value J, we have $\zeta_k = \hat{u}^{-k} J^k \stackrel{\triangle}{=} u^{-k}$ with $u = \frac{\hat{u}}{J}$. Since, on average, at each level, each off period accounts for the $\frac{m_{off}^i}{m_{on}^i + m_{off}^i}$ fraction of the total time, and the remaining time is divided into $\hat{u}$ on periods at the next level, each on period has an average length of m_{on}^{i+1}. Therefore, the average length of an on period at level k is ζ_k (given in Eq. (3.45)). Altogether, there are u^k such periods in which traffic will be transmitted. Next, the *Hölder* exponents can be approximated by calculating the order of changing of the signal during these periods.

Approximate of *Hölder* exponent The degree of local irregularity of a signal X and its singularity structure at a given time point t_0 can be characterized by an algebraic function $\alpha(t_0)$ which is the largest value of α such that:

$$|X(t') - X(t_0)| \leq C|t' - t_0|^{\alpha}, \tag{3.46}$$

for all t' sufficiently close to t_0. Let

$$\alpha_k(t) = \alpha_{n_k}^k := -\frac{1}{k} \log_u |X((n_k + 1)u^{-k}) - X(n_k u^{-k})|. \tag{3.47}$$

Note that Eq. (3.47) is obtained by taking log operation on both sides of Eq. (3.46) and letting $t' = (n_k + 1)u^{-k}$ and $t_0 = n_k u^{-k}$. Obvious, Eq. (3.47) is an approximation of Eq. (2.19). Thus

$$\alpha(t) = \lim_{k \to \infty} \alpha_k(t) \tag{3.48}$$

can be used as an approximation of the *Hölder* exponent at time t, where $t \in (n_k u^{-k}, (n_k + 1)u^{-k})$, i.e., t belongs to the n_k^{th} interval at level k.

Multifractal spectrum $g(\alpha)$ and partition function $\tau(q)$ We now quantify the values of the limiting scaling exponent $\alpha(t)$ that appear in a signal and how often these values are encountered. This is captured in the multifractal spectrum $d_x(a)$ — the Hausdorff dimension of the set $\{t : \alpha(t) = \alpha\}$. As it is usually difficult to derive the multifractal spectrum directly, we instead consider the partition function

$$\tau(q) = \lim_{k \to \infty} -\frac{1}{k} \log_u S_k(q), \tag{3.49}$$

where $q > 0$ and $S_k(q)$ is called the structure function and is given by:

$$S_k(q) = \sum_{n=0}^{u^k - 1} |X((n + 1)u^{-k}) - X(nu^{-k})|^q = \sum_{n=0}^{u^k - 1} u^{-qk\alpha_n^k}. \tag{3.50}$$

Let $g_k(\alpha, \epsilon) \stackrel{\triangle}{=} dim(\alpha_k(t) \in (\alpha - \epsilon, \alpha + \epsilon))$ and the "histogram" $g(\alpha)$ be defined as

$$g(\alpha) := \lim_{\epsilon \to 0} \lim_{n \to \infty} \frac{1}{k} \log_u g_k(\alpha, \epsilon). \tag{3.51}$$

Combining Eqs. (3.50) and (3.51), $S_k(q)$ can be further expressed as

$$S_k(q) := \sum_{\alpha} \sum_{\alpha_n^k \simeq \alpha} u^{-kq\alpha} \simeq \sum_{\alpha} u^{-k(q\alpha - g(\alpha))} \simeq u^{-k \inf_\alpha (q\alpha - g(\alpha))}. \tag{3.52}$$

Hence we have

$$\tau(q) := \inf_{\alpha}(q\alpha - g(\alpha)). \tag{3.53}$$

Demonstration of the concavity of $\tau(q)$ To find $\tau(q)$, we need to calculate $S_k(q)$. Let $M_n^k = |X((n+1)u^{-k}) - X(nu^{-k})|$, $n = 0, ..., u^k - 1$ be identically distributed. Then we can calculate $E(S_k(q))$ as

$$E(S_k(q)) = u^k \cdot E(M_n^k)^q = u^k \cdot \prod_{i=0}^{k-1} E(M_n^i)^q, \tag{3.54}$$

where the second equality results from $M_n^k = \prod_{i=0}^{k-1} M_n^i$. Usually wavelet based tools are used to measure $S_k(q)$ and hence $\tau(q)$. To give a qualitative demonstration, we consider a simple case to illustrate that the proposed hierarchical model exhibits multifractality. We assume M_n^i follows the uniform distribution in $(0, b)$ [3] with $b = \frac{m_{on}+m_{off}}{m_{on}}$ for all $0 \le i \le k$, and hence $E(M_n^i)^q = \frac{b^q}{q+1}$. Consequently, we have

$$E(S_k(q)) = u^k \frac{b^{qk}}{(q+1)^k} = u^{-k\tau(q)}, \tag{3.55}$$

and hence

$$\tau(q) = \log_u(q+1) - q \log_u b - 1. \tag{3.56}$$

That is, $\tau(q)$ is the Legendre transform of $g(\alpha)$ and is concave. As shown in Fig. 3.12, $\tau(q)$ exhibits nonlinear (concave) behaviors, which indicates multifractality.

To validate the multifractal property of the hierarchical model, we carry out simulation that generates traffic in compliance with the proposed model. In the simulation, we set $\alpha_N = 2.43$, $\alpha_b = 1.3$, $\alpha_i = 1.5$, $\alpha_I = 1.4$, and therefore $\alpha_B = 1.3$. We then use a wavelet based tool provided by Abry *et al.* [111] to calculate the partition function of a signal, and depict the calculated results of $\tau(q)$ in Fig. 3.13 (a). The generated traffic does has a concave $\tau(q)$ function. We also show that the real Internet traces available at *IRCache.net* [72] possess the multifractal property. In particular,

[3] We do not claim that M_n^i is uniformly distributed in reality. Instead we only use this simple example to show the concavity of $\tau(q)$.

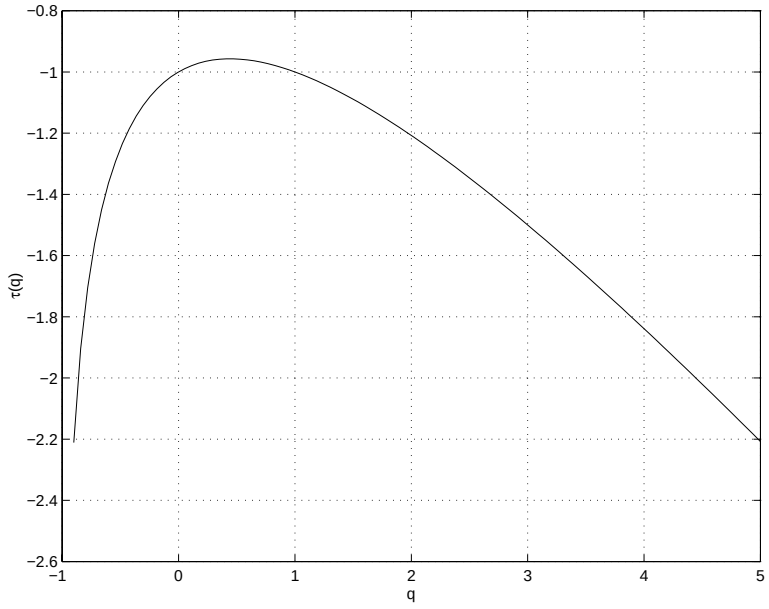


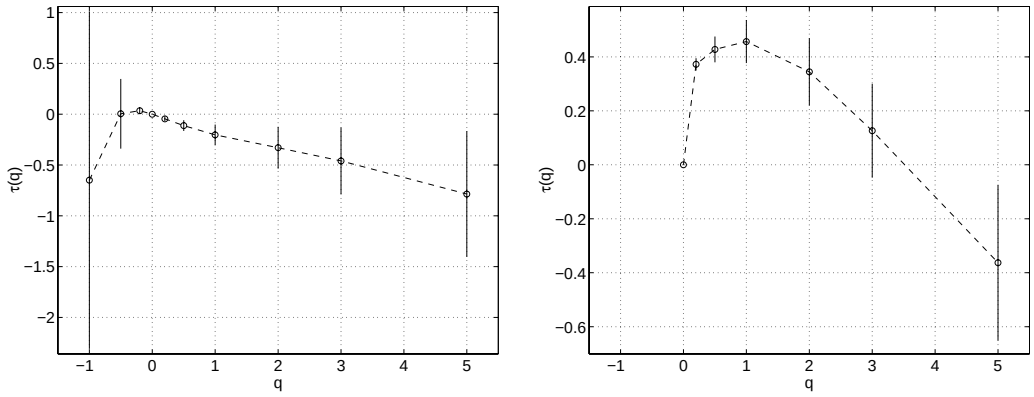
Figure 3.12: The partition function $\tau(q)$ versus q.

we calculate the partition function of the traces given in Section 3.1.3 and give a typical result in Fig. 3.13 (b). The partition function of the real traces is also concave. Although the two figures in Fig. 3.13 do not resemble each other, they do share one common feature — both of them are concave, thus indicating both the generated traffic and the real Internet traffic are multifractal.

3.4 Queuing Behavior Under the Hierarchical Model

Queuing analysis is always an important topic in network traffic control and resource provisioning. Existing work [97], [18] has shown that self-similar or LRD processes can cause much heavier tail in content process than a traditional Poisson input process. Due to the uniqueness of the hierarchical model proposed in this chapter, we will study the queuing behavior of a single server queue (SSQ) under the proposed hierarchical model. Not to our surprise, our finding is consistent with the existing results.

In Section 3.4.1 we study the queuing behavior of SSQ with a single input flow, and we explore the multiple input flows case in Section 3.4.2.



(a) Generated traffic (b) Real trace

Figure 3.13: Partition functions of model generated traffic and real traffic trace.

3.4.1 Single Input Flow

To investigate the queuing behavior of a SSQ that is fed with the multifractal traffic generated by the hierarchical model, we consider the two-tier hierarchical model, in which the renewal process cycles through alternating ON and OFF periods and each ON period, in turn, consists of several level-1 on and off periods. We use this model as the input to a fluid SSQ, and derive the tail distribution of the content process, $C(t)$, of the queue. Specifically, we first derive the content process at the ending points of OFF periods, i.e., the renewal points of an alternating renewal process (see Section. 2.5). Then we apply Cohen's theorem [12] (summarized in Section. 2.5) to obtain the asymptotic behavior of the content process as time goes to infinity. Finally, we extend the result to any time point, and derive the asymptotic behavior of the content process as t goes to infinity.

Content process at renewal points and its asymptotic behavior Without loss of generality, we assume the traffic is generated at the rate of 1 in an on period. We represent the renewal sequence as $\{S_n, n \geq 0\}$ with $S_n = S_0 + \sum_{i=1}^{n}(B_i + I_i)$, $n \geq 1$, where $\{B_i, i \geq 1\}$ represents ON periods and $\{I_i, i \geq 1\}$ represents OFF periods. S_0 is a random variable chosen to make S_n a stationary process and consists of an on period B_0 and an off period I_0. The interested reader is referred to [65] for a

detailed account of S_0. Each ON period B_i can be further expressed as

$$B_i = \sum_{j=1}^{N_i} (b_{i,j} + i_{i,j}), \tag{3.57}$$

where $b_{i,j}$ and $i_{i,j}$ represent the j^{th} on and off periods respectively in B_i, and N_i is the number of level-1 on periods in B_i. Then, S_n can be rewritten as

$$S_n = \sum_{i=0}^{n} (\sum_{j=1}^{N_i} (b_{i,j} + i_{i,j}) + I_i). \tag{3.58}$$

Let $\{\gamma_t, t \geq 0\}$ be the indicator process that is 1 in an on period and 0 otherwise. The cumulative multifractal input generated by the hierarchical model up to time t can be expressed as

$$a_t \stackrel{\triangle}{=} \int_0^t \gamma_u du. \tag{3.59}$$

By the definition in Section 3.1, we have $a_t \approx t \cdot \frac{N \cdot m_{on}}{m_{ON}+m_{OFF}}$, where $m_{ON} = N(m_{on} + m_{off})$ and $N = E(N_i)$ given $\{N_i\}$ is i.i.d.. The long term input rate is $\frac{N m_{on}}{m_{ON}+m_{OFF}}$. Suppose the service rate of the system when the amount of accumulated traffic is x is

$$r_s(x) = \begin{cases} r, & \text{if } x > 0, \\ 0, & \text{if } x \leq 0, \end{cases} \tag{3.60}$$

and the traffic generation rate does not exceed the service rate, i.e.,

$$\frac{N m_{on}}{m_{ON} + m_{OFF}} < r < 1, \tag{3.61}$$

then the content process $\{C(t), t \geq 0\}$ satisfies the following relation

$$dC(t) = da_t - r_s(C(t))dt. \tag{3.62}$$

That is, in an on period traffic enters the system at a net rate of $1 - r$ and in an off period traffic is served (and consumed) at a rate of r. The process $\{C(t), t \geq 0\}$ can be shown to be a regenerative

process with the regeneration times

$$\{D_n\} \stackrel{\triangle}{=} \{S_n : C(S_n-) = 0\}. \tag{3.63}$$

Note that the regenerative time points are the time instants when the content process becomes empty and the input process commences to fill the queue. Since $\{S_n\}$ is a stable renewal process, and by Smith's Theorem [107] (Section. 2.5), both $\{C(S_n), n \geq 0\}$ and $\{C(t), t \geq 0\}$ have limit distributions.

To derive the limit distributions of $\{C(S_n), n \geq 0\}$ and $\{C(t), t \geq 0\}$, we express the total length of an ON period till the n^{th} recursion as

$$S_n^B = \sum_{i=0}^{n} B_i = \sum_{i=0}^{n} (\sum_{j=1}^{N_i} (b_{i,j} + i_{i,j})), \tag{3.64}$$

and the total length of an OFF period till the n^{th} recursion as

$$S_n^I = \sum_{i=1}^{n} I_i. \tag{3.65}$$

Let $\Omega_n \stackrel{\triangle}{=} (1-r)\Omega_n^b - r\Omega_n^i$, with $\Omega_n^b = \sum_{i=0}^{n} \sum_{j=1}^{N_i} b_{i,j}$ and $\Omega_n^i = \sum_{k=0}^{n} \sum_{j=1}^{N_k} i_{k,j}$. As $C(t)$ increases with rate $1-r$ during an on period, we define the index of empty instant T_s to be

$$T_s = \inf_n(\Omega_n - rS_n^I \leq 0) = \inf_n((1-r)\Omega_n^b - r_s(\Omega_n^i + S_n^I) \leq 0). \tag{3.66}$$

If $T_s = t$, we have

$$(1-r)\Omega_j^b - r_s(\Omega_j^i + S_n^I) \begin{cases} > 0, & \text{if } 1 \leq j < t, \\ \leq 0, & \text{if } j = t. \end{cases} \tag{3.67}$$

Let 1_{cond} be an indicator function, i.e., $1_{cond} = 1$ if *cond* holds; $= 0$ otherwise. By Smith's Theorem, we have for $x > 0$:

$$\overline{U}(x) = 1 - U(x) \stackrel{\triangle}{=} P_r(C(S_n) > x) \rightarrow \frac{E(\sum_{j=1}^{T_s} 1_{C(S_j)>x})}{E(T_s)}, \tag{3.68}$$

and

$$\overline{V}(x) = 1 - V(x) \stackrel{\triangle}{=} P_r(C(t) > x) \rightarrow \frac{E(\int_{j=1}^{D_1} 1_{C(u)>x} du)}{E(S_{T_s})}, \tag{3.69}$$

where by Wald's equation $E(S_{T_s}) = (m_{ON} + m_{OFF})E(T_s)$. To derive $U(x)$, we have the iterative relationship between $C(S_n)$ and $C(S_{n+1})$ as

$$\begin{aligned} C(S_{n+1}) &= (C(S_n) + \sum_{j=1}^{N_{n+1}} ((1-r)b_{n+1,j} - ri_{n+1,j}) - rI_{n+1})^+ \\ &= (C(S_n) + \eta_{n+1})^+, \end{aligned} \tag{3.70}$$

where$\{\eta_n = \sum_{j=1}^{N_n}((1-r)b_{n,j} - ri_{n,j}) - rI_n\}$ is a sequence of i.i.d random variables with

$$E(\eta_n) = N(1-r)m_{on} - Nrm_{off} - rm_{OFF} = Nm_{on} - r(m_{ON} + m_{OFF}). \tag{3.71}$$

Note that we use the relation $m_{ON} = N(m_{on} + m_{off})$ in Eq. (3.71). Eq. (3.70) can be further written as $C(S_{n+1}) = (((C(S_0) + \eta_1)^+ + \eta_2)^+ + ... + \eta_{n+1}))^+$, i.e., assuming $C(0) = 0$ we have

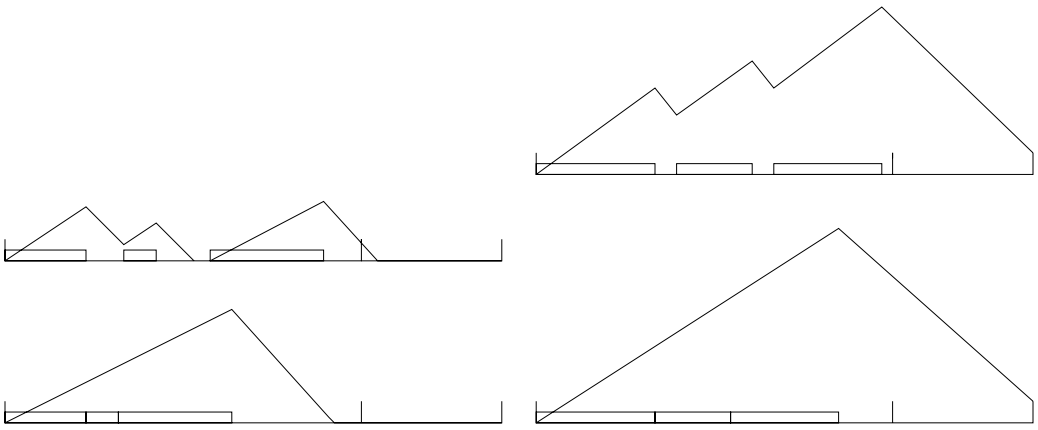
$$C(S_n) \sim \max_{0 \le n < \infty} \sum_{i=1}^{n} \eta_i < \infty. \tag{3.72}$$

Note that as we only care about the time points $\{S_n\}$, we can move all the level-1 on periods together in each ON period. Let the content process corresponding to the shuffled input model be $\hat{C}(S_n)$. The following Lemma 2 states that $C(S_n) = \hat{C}(S_n)$

Lemma 2: The content processes at time $\{S_n, n \ge 0\}$ are the same under the original and shuffled input models, i.e.,

$$C(S_n) = \hat{C}(S_n). \tag{3.73}$$

Proof: It is sufficient to prove the first cycle that starts at $C(0) = \hat{C}(0) = 0$. We show two cases in Fig. 3.14. Under both the original and shuffled input models, the cycle time is the same and can be denoted as $T = T_{ON} + T_{OFF}$. T_{ON} can in turn be expressed as $T_{ON} = T_{on} + T_{off}$ where T_{on} is the summation of all the level-1 on periods and T_{off} the summation of all the level-1 off periods. Therefore, $T = T_{on} + T_{off} + T_{OFF}$. The only difference between the two models are the positions



(a) Case 1: content is empty at S_n (b) Case 2: content > 0 at S_n

Figure 3.14: Two cases of the queue content process under the original and shuffled input models.

of these level-1 on and off periods. We have

$$C(S_1) = 1 \cdot T_{on} - r \cdot (T_{off} + T_{OFF}) = \hat{C}(S_1), \tag{3.74}$$

regardless of the positions of these on and off periods.

We are now in a position to characterize $\overline{U}(x)$.

Theorem 4: Given the distributions of P and M in Section 3.1, and

$$\overline{P}(x) \sim x^{\alpha_b} L(x), 1 < \alpha_b < 2, x \to \infty, \tag{3.75}$$

and

$$\overline{M}(x) \sim x^{\alpha_N} L(x), 1 < \alpha_N < 2, x \to \infty, \tag{3.76}$$

we have

$$\overline{U}(x) \sim \xi x^{-(\alpha_{min}-1)} L(x), x \to \infty, \tag{3.77}$$

where $\xi = \frac{\rho}{1-\rho} \cdot \frac{(1-r)^{\alpha_{min}-1}}{(\alpha_{min}-1)Nm_{on}}$, $\rho = \frac{Nm_{on} \cdot (1-r)}{(Nm_{off}+m_{OFF})r} < 1$ is the traffic intensity, and $\alpha_{min} = min(\alpha_b, \alpha_N)$.

Proof: By Lemma 2, Since $C(S_n) = \hat{C}(S_n)$, it suffices to consider the content process, $\{\hat{C}(S_n), n \geq 0\}$, under the shuffled input model. Thus we have a new ON'/OFF' process, with ON' equal to the summation of several level-1 on periods, and OFF' equals the summation of OFF and sev-

eral level-1 off periods. Following the same line of arguments in Section 3.2.1 we can show the distribution of ON', $F_{ON'}(x)$, follows the rule:

$$1 - F_{ON'}(x) \sim x^{-\alpha_{min}} L(x), \text{as } x \to \infty, \tag{3.78}$$

where $\alpha_{min} = min(\alpha_b, \alpha_N)$, and $L(x)$ is a slowly varying function. In order to apply the Cohen's theorem [12] (Section. 2.5) (page 387) we fit the new ON'/OFF' model to a $GI/G/1$ queuing model with the traffic density

$$\rho = \frac{Nm_{on} \cdot (1-r)}{(Nm_{off} + m_{OFF})r} < 1, \tag{3.79}$$

and the corresponding service time distribution in the $GI/G/1$ queue, $B(x)$, has the following relation with $F_{ON'}(x)$

$$B(x) \sim F_{ON'}((1-r)x). \tag{3.80}$$

Directly apply the Cohen's theorem (Section. 2.5) [12] (page 387) we have

$$\overline{U}(x) \sim \frac{\rho}{1-\rho} \cdot \frac{(1-r)^{\alpha_{min}-1}}{(\alpha_{min}-1)Nm_{on}} \cdot x^{-(\alpha_{min}-1)} L(x), \tag{3.81}$$

where $\alpha_{min} = min(\alpha_b, \alpha_N)$.

Note that Eq. (3.81) indicates that $C(t)$ exhibits a power law behavior at regenerative time points. In what follows we will derive $\overline{V}(x)$, as $V(x)$ is more general than $U(x)$. We define the Laplace transform of $V(x)$ and $U(x)$ to be $V(s)$ and $U(s)$ respectively.

Derivation of $\overline{V}(x)$ To derive $\overline{V}(x)$, we express the amount of time during which the process $C(t)$ goes beyond x in the n^{th} ON/OFF cycle as

$$L_n(x) = \int_{S_n}^{S_{n+1}} 1_{(C(t)>x)} dt. \tag{3.82}$$

Recall that we have defined the sequences $\{B_n\}$, $\{I_n\}$, and $\{C(S_n)\}$. We further define the sequences $\{b_n^m\}$, $\{i_n^m\}$ and $\{s_n^m\}$ for the level-1 on, off and regenerative sequences in an ON period respectively (Fig. 3.15), and consider the corresponding stationary sequences $\{b_n^m\}$,$\{i_n^m\}$ and $\{s_n^m\}$ by carefully selecting the distribution of the starting on period [65]. With the definition of these stationary

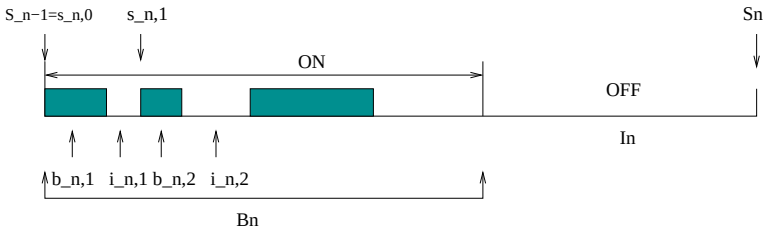


Figure 3.15: Decomposing the time interval (S_{n-1}, S_n).

sequences, we now verify that $\{L_n(x)\}$ is also a stationary sequence.

Suppose there are N_n on/off periods in an ON period, then, if $S_n \le t < S_n + B_{n+1}$, we have, for all $1 \le m \le N_n$, $S_{n-1} = s_n^0$, $S_n + B_{n+1} = s_{n+1}^{N_{n+1}}$, and

$$C(x) = \begin{cases} C(s_n^m) + (1-r)(t - s_n^m), \\ \quad \text{if } s_n^m \le t < s_n^m + b_n^{m+1}, \\ (C(s_n^m + (1-r)b_n^{m+1} - r_s(t - (s_n^m + b_n^{m+1})))^+, \\ \quad \text{if } s_n^m + b_n^{m+1} \le t < s_n^{m+1}. \end{cases} \tag{3.83}$$

On the other hand, if $s_{n+1}^{N_{n+1}} \le t < S_{n+1}$, we have

$$C(x) = (X(s_n^{N_n}) - r_s(t - (S_n + B_{n+1})))^+. \tag{3.84}$$

Then we can rewrite $L_n(x)$ as

$$\begin{aligned} L_n(x) &= \int_{S_n}^{S_{n+1}} 1_{(x,\infty)}(C(t))dt \\ &= \int_{s_n^1}^{s_n^1+b_n^2} + \int_{s_n^1+b_n^2}^{s_n^2} + \ldots + \int_{s_n^{N_n-1}}^{s_n^{N_n-1}+b_n^{N_n}} + \int_{s_n^{N_n-1}+b_n^{N_n}}^{s_n^{N_n}} + \int_{s_n^{N_n}}^{S_{n+1}} \\ &= \sum_{j=1}^{N_n-1} \Big(\int_0^{b_n^j} 1_{x,\infty}((C(s_n^j) + (1-r))du + \\ &\quad \int_0^{i_n^j} 1_{x,\infty}((C(s_n^j) + (1-r)b_n^{j+1} - ru)^+)du) + \\ &\quad \int_0^{I_{n+1}} 1_{x,\infty}((C(S_n) + (1-r)B_{n+1} - ru)^+)du \\ &\triangleq f(C(B_n), C(b_n^m), B_n, I_n, b_n^m, i_n^m N_n), 1 \le m \le N_n. \end{aligned}$$

Note that the above is an instantaneous function of a set of stationary sequences, therefore we verify that $\{L_n(x)\}$ is also a stationary sequence. Suppose the random variable $C(\infty)$ has the distribution V, with the stationarity of $\{L_n(x)\}$, we have (by Smith's Theorem [107], Eq. (2.23))

$$P_r(C(\infty) \in A) = m^{-1} E \int_0^{S_1} 1_{C(s)\in A} ds, m = m_{ON} + m_{OFF} < \infty. \tag{3.85}$$

Therefore we have

$$P_r(C(\infty) > x) = m^{-1} E \int_0^{S_1} 1_{C(s)\in(x,\infty)} ds. \tag{3.86}$$

To obtain the distribution of V, we consider the Laplace transform of $C(\infty)$:

$$\begin{aligned} E(e^{-sC(\infty)}) &= \frac{1}{m} E \int_0^{S_1} e^{-sC(u)} du = \frac{1}{m} N \cdot E(\int_0^{b_1^1} + \int_0^{i_1^1}) + \frac{1}{m} E \int_0^{I_1} \\ &= \frac{1}{m} N \cdot (E e^{-sC(0)} (\frac{1 - e^{-s(1-r)b_1^1}}{s(1-r)}) + E \int_0^{i_1^1} e^{-s(C(0)+(1-r)b_1^1 - rt)^+} dt) \\ &\quad + \frac{1}{m} E \int_0^{I_1} e^{-s(C(0)+(1-r)B_1 - rt)^+} dt. \end{aligned} \tag{3.87}$$

Let E_1, E_2 and E_3 denote the three terms in Eq. (3.87). Then we have

$$E_1 = \frac{N}{m} U(s) \cdot \frac{1 - P(s(1-r))}{s(1-r)}, \tag{3.88}$$

where $P(s)$ is the Laplace transform of an level-1 on period (Section. 3.2.1). E_2 can be rewritten as

$$\begin{aligned} E_2 &= \frac{N}{m} \cdot (E \int_0^{\infty} e^{-s(C(0)+(1-r)b_1^1 - rt)^+} 1_{t \le i_1^1, rt \le C(0)+(1-r)b_1^1} dt \\ &\quad + E \int_0^{\infty} 1_{t \le i_1^1, rt > C(0)+(1-r)b_1^1} dt) \\ &= \frac{N}{m} E e^{-s(C(0)+(1-r)b_1^1)} \frac{e^{sr_s(i_1^1 \wedge (\frac{C(0)+(1-r)b_1^1}{r}))} - 1}{sr} \\ &\quad + \frac{N}{m} E(i_1^1 - \frac{C(0) + (1-r)b_1^1}{r})^+ \\ &\triangleq \frac{N}{m}(E_{2a} + E_{2b}). \end{aligned}$$

Note that E_{2b} is independent of s. E_{2a} can be expressed as

$$E_{2a} = \frac{1}{sr} E(e^{-s(C(0)+(1-r)b_1^1 - ri_1^1)^+} - e^{-s(C(0)+(1-r)b_1^1)}). \tag{3.89}$$

Note $(C(0)+(1-r)b_1^1 - ri_1^1)^+ = C(s_1^1)$, and $\{s_n^m\}$ is an embedded regenerative process. Let $u(x)$ be the CDF of $C(s_n^m)$. Similarly, by Cohen's theorem [12] (Section. 2.5), and using the same technique as in the derivation of Eq. (3.81), we obtain

$$1 - u(x) \sim \xi_s x^{-(\alpha_b - 1)} L(x), x \to \infty \tag{3.90}$$

and $\xi_s = \frac{\rho_s}{1-\rho_s} \cdot \frac{(1-r)^{\alpha_b - 1}}{(\alpha_b - 1)m_b}$, and $\rho_s = \frac{m_b(1-r)}{m_i r}$. Let the Laplace transform of $u(x)$ be denoted as $u(s)$. Then Eq. (3.89) can be written as

$$E_{2a} = \frac{1}{sr} E(e^{-s(C(s_1^1))} - e^{-s(C(0)+(1-r)b_1^1)}) = \frac{1}{sr}(u(s) - U(s)P((1-r)s)).$$

For notational convenience, we also denote $E_{2b} = \kappa_0$.

Similarly, E_3 can be expressed as

$$E_3 = \frac{1}{m}(\frac{U(s) - U(s)G((1-r)s)}{sr}) + \kappa_1, \tag{3.91}$$

where $G(s)$ is the Laplace transform of $G(x)$ (Section. 3.2.1) and $\kappa_1 = E(I_1 - \frac{C(0)+(1-r)B_1}{r})^+$. Thus we have

$$V(s) = Ee^{-sC(\infty)} = \frac{U(s)}{m}(N \cdot \frac{r - P((1-r)s)}{s(1-r)r} + \frac{1 - G((1-r)s)}{sr}) + \frac{N}{srm}u(s) + \kappa \tag{3.92}$$

where $\kappa = \frac{1}{m} \cdot (N\kappa_0 + \kappa_1)$, and $V(0) = \lim_{s\to\infty} V(s) = \kappa$. As shown in [8], α_N is always less than α_b, and hence Eq. (3.81) and Eq. (3.90) give $u(x) \sim U(x)$, as $x \to \infty$, and $u(s) \sim U(s)$, as $s \to 0$. In analyzing the asymptotic behavior of $V(x)$, we approximate $u(s)$ as $U(s)$, i.e., $u(s) \approx U(s)$, and $V(s)$ can be further approximately simplified as:

$$V(s) \approx \frac{N}{m}U(s)(\frac{1 - P((1-r)s)}{sr_s(1-r)} + \frac{1 - G((1-r)s)}{Nsr}) + \kappa. \tag{3.93}$$

Then we take the inverse Laplace transform of Eq. (3.93) and obtain

$$V(x) \approx U(x) \star Z(x) + \kappa, \tag{3.94}$$

where $Z(x)$ has Laplace transform $Z(s) = \frac{N}{m}(\frac{1-P((1-r)s)}{sr_s(1-r)} + \frac{1-G((1-r)s)}{Nsr})$. Hence we have

$$\begin{aligned} 1 - V(x) &\approx 1 - \kappa - U(x) \star Z(x) = (1-\kappa)(1 - \frac{U(x) \star Z(x)}{1-\kappa}) \\ &= (1-\kappa)(1 - U(x) * M(x)), \end{aligned} \tag{3.95}$$

where $M(x) = \frac{Z(x)}{1-\kappa}$ and has Laplace transform $M(s) = \frac{Z(s)}{1-\kappa}$. Now we need to check whether or not M is a valid probability measure, i.e. $M(s)|_{s=0} = 1$. Letting $s \to 0$ in Eq. (3.93) we obtain

$$1 = \frac{N}{m}(\frac{m_b}{r} + \frac{(1-r)m_B}{Nr}) + \kappa. \tag{3.96}$$

Hence we have

$$1 - \kappa = \frac{Nm_b + (1-r)m_B}{mr}. \tag{3.97}$$

After some algebraic operations, it is straightforward to obtain

$$M(s)|_{s=0} = \frac{m_b + \frac{1-r}{N}m_{ON}}{m_b + (1-r)(m_b + m_i)} = 1 \tag{3.98}$$

based on the fact that $m_{ON} = N(m_b + m_i)$.

Thus $M(s)$ can be further expressed as

$$M(s) = \frac{1 - P((1-r)s)}{s(1-r)a} + \frac{1 - G((1-r)s)}{s(1-r)b}, \tag{3.99}$$

where $a = \frac{(1-\gamma)rm}{N} = m_b + (1-r)(m_b + m_i)$, $b = \frac{(1-\gamma)mr}{(1-r)} = \frac{Nr}{1-r}a$. $M(s)$ can also be expressed as

$$\begin{aligned} M(s) &= \int_0^\infty e^{-s(1-r)x}\frac{1-P(x)}{a}dx + \int_0^\infty e^{-s(1-r)x}\frac{1-G(x)}{b}dx \\ &= \int_0^\infty e^{-sy}\frac{1-P(\frac{y}{1-r})}{a(1-r)}dy + \int_0^\infty e^{-sy}\frac{1-G(\frac{y}{1-r})}{b(1-r)}dy. \end{aligned} \tag{3.100}$$

This implies that the probability density function of M is $m(x) = \frac{1-P(\frac{x}{1-r})}{a(1-r)} + \frac{1-G(\frac{x}{1-r})}{b(1-r)}$. Based on the fact that: $1 - P(x) \sim x^{-\alpha_b}L(x)$, and $1 - G(x) \sim x^{-\alpha_B}L(x)$, as $x \to \infty$, we have

$$\begin{aligned} 1 - M(x) &= \int_x^\infty m(y)dy \\ &\sim \int_x^\infty \frac{(\frac{y}{1-r})^{-\alpha_b}}{a(1-r)}L(y)dy + \int_x^\infty \frac{(\frac{y}{1-r})^{-\alpha_B}}{b(1-r)}L(y)dy \\ &\sim \frac{x^{-(\alpha_b-1)}L(x)}{(1-r)^{1-\alpha_b}(\alpha_b-1)a} + \frac{x^{-(\alpha_B-1)}L(x)}{(1-r)^{1-\alpha_B}(\alpha_B-1)b}. \end{aligned} \tag{3.101}$$

In Section 3.2.1, we know $\alpha_B = min(\alpha_b, \alpha_i, \alpha_N) \leq \alpha_b$. Hence the second term dominates between the two items in Eq. (3.101). By plugging Eq. (3.101) in Eq. (3.95) we have

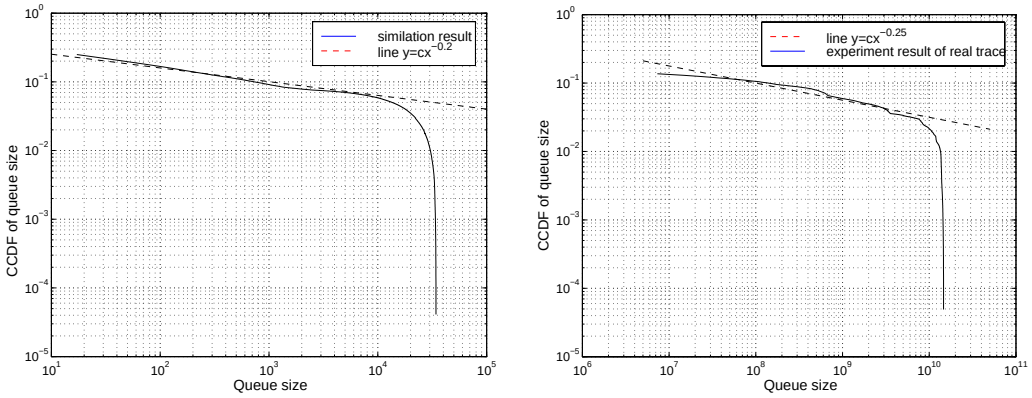
$$\begin{aligned} 1 - V(x) &\approx (1-\gamma)(1 - U(x) \star M(x)) \\ &\sim \frac{Nm_b + (1-r)m_B}{mr} \cdot (\xi + \frac{1}{(1-r)^{1-\alpha_b}(\alpha_b-1)a} \\ &\quad + \frac{1}{(1-r)^{1-\alpha_B}(\alpha_B-1)b})x^{-(\alpha_B-1)}L(x) \\ &= \zeta x^{-(\alpha_B-1)}L(x). \end{aligned} \tag{3.102}$$

Eq. (3.102) states

(i) The asymptotic tail distribution of the content process follows a power law rather than an exponential law. This is an index of heavy-tailedness of the content.

(ii) The tail of the content process is determined by the shape parameter, α_B, of the ON period. As the heavy-tailed characteristic of an ON period is determined by the shape parameters of its components, i.e., $\alpha_B = min(\alpha_b, \alpha_i, \alpha_N)$, the tail behavior of the content process is determined by the heaviest component of an ON period. A single level ON/OFF model (mono-fractal model) does not possess this property. Therefore, when using a single level ON/OFF model, it is critical to correctly characterize its ON period distribution. If the shape parameter of the ON period distribution does not catch the heaviest component in the real traffic, the model may render a much lighter tail behavior of a SSQ system.

To validate the derivation results, we have carried out the following experiments. We use the hierarchical model to generate input traffic. The shape parameters are set to $\alpha_b = 1.2, \alpha_i =$



(a) CCDF of model traffic (b) CCDF of real trace

Figure 3.16: The CCDF of the queue length of a SSQ with model generated traffic and real trace traces as input.

$1.5, \alpha_I = 1.4$, and $\alpha_N = 2$, and hence $\alpha_{min} = min(\alpha_b, \alpha_i, \alpha_I, \alpha_N) = 1.2$. The service rate of the SSQ system r is set to 0.8 (which is greater than the input rate $\frac{Nm_i}{m} \approx 0.6$). Fig. 3.16(a) depicts the simulation results of the CCDF of the content process in a log-log plot. As shown in Fig. 3.16 (a), the CCDF coincides very well the line $y = cx^{-(\alpha_b - 1)} = cx^{-0.2}$. Note that the big slump when the queue size exceeds 10^5 is due to the physical limitation of CPU and memory, since in the experiments we cannot generate a queue with queue size $\to \infty$.

We have also carried out experiments using real Internet traces gathered from *IRCache.net* [72]. We first fit the traces into our two-tier hierarchical model, and obtain the four parameters $\alpha_b = 1.5$, $\alpha_i = 1.2$, $\alpha_I = 1.9$, $\alpha_N = 1.8$. As a result $\alpha_{min} = 1.20$. The long term input rate is $20kbps$. Then, we use the traces as the input to a fluid SSQ with service rate $r = 25kbps$. Fig. 3.16 (b) shows the tail distribution of the queue length. As shown in Fig. 3.16 (b), the CCDF coincide very well the line $y = cx^{-0.25} \approx cx^{-(\alpha_{min} - 1)}$.

3.4.2 Multiple Input Flows

In this section we study the asymptotic queuing behavior of a fluid SSQ with multiple input flows generated by the proposed hierarchical model, especially when the number of flows $N_f \to \infty$. We know that the aggregate traffic of N_f flows cannot be modeled as a renewal or regenerative process. To obtain the asymptotic queuing behavior, we first show that as $N_f \to \infty$, the aggregate input traffic converges weakly to a limiting Gaussian process. Then we derive the content process $(C(t))$

and its tail distribution.

Gaussian property of the aggregate input traffic We start our study from the Theorem 5

Theorem 5: Let F be a stationary renewal process, with $F(0,t)$ representing the number of renewal points within time interval $(0,t)$, $t \geq 0$ and with finite and nonzero mean intensity λ_f. Denoting by $\Pi(t)$ the cumulative distribution function (CDF) of the inter-renewal times, and we assume that: $\pi(t) = \frac{d\Pi(t)}{dt}$ exists and is bounded in some neighborhood of 0. Then: given a sequence $\{F_i\}_{i\geq 0}$ of i.i.d. copies of F, the sequence of processes $\{F^{N_f}\}_{N_f>0}$ defined by :

$$F^{N_f}(t) = \frac{1}{\sqrt{N_f}} \sum_{i=1}^{N_f} (F_i(0,t) - \lambda_f t), t \geq 0, \tag{3.103}$$

converges weakly to a limiting Gaussian process with a.s. continuous paths as $N_f \to \infty$. *Proof: see book [100](chapter 5)*

Here $F \in D(R^+)$, where $D(R^+)$ denotes the space of cadlag function on R^+ (left continuous function with right hand limits defined on real positive domain), and is endowed with the topology of convergence in the J_1-Skorokhod topology on bounded intervals. Following Theorem 5, similar results for process $\{\Gamma_t^N\}$ and $\{A_t^N\}$ defined in Section 3.2.2 can be obtained: we define the sequence process Θ^N and Φ^N to be:

$$\Theta^N = \frac{\Gamma_t^N - N\mu\mu_h}{\sqrt{N}}, t \geq 0 \tag{3.104}$$

and

$$\Phi^N = \frac{A_t^N - N\mu\mu_h t}{\sqrt{N}}, t \geq 0, \tag{3.105}$$

which are normalized sequence of $\{\Gamma_t^N\}$ and $\{A_t^N\}$. Both Θ^N and Φ^N converge weakly in the space of $D(R^+)$ toward a zero-mean stationary(stationary increments)continuous Gaussian process as $N \to \infty$.

Queue content process and its tail distribution We still use $C(t)$ to denote the queue content at time t. The input process is A_t^N, and the service rate is $r > 0$ if $C(t) \geq 0$ and $r = 0$ otherwise. Then we have:

$$C(t) = (A_t^N - \int_0^t r_s(t)dt)^+. \tag{3.106}$$

We define the queue idle time to be $T_{idle}(t)$, and it can be expressed as:

$$T_{idle}(t) = \int_0^t 1_{C(u)\leq 0} du. \tag{3.107}$$

Then the queue content distribution is endowed with the equation:

$$P_r(C(t) \leq x) = P_r(A_t^N - rt \leq x) - P_r(A_t^N - rt \leq x \leq A_t^N - rt + T_{idle}(t)). \tag{3.108}$$

According to [10] the second item in Eq. 3.108 equals:

$$\begin{aligned} & P_r(A_t^N - rt \leq x \leq A_t^N - rt + T_{idle}(t)) \\ = & \frac{\partial}{\partial x} \int_0^t P(A_t^N - A_u^N - rt + ru \leq x, C(u) = 0) du \\ = & \frac{\partial}{\partial x} \int_0^t P((A_t^N - rt) - (A_u^N - ru) \leq x | C(u) = 0) P(C(u) = 0) du. \end{aligned}$$

Let $\Upsilon(t) = A_t^N - rt$. We assume the stationarity of $\Upsilon(t)$. Notice $P(C(u) = 0)$ is the probability that the queue is idle. With input rate $N\mu\mu_h \to \lambda$, as $N \to \infty$, we have $P(C(u) = 0) = 1 - \frac{\lambda}{r}$. Eq. 3.108 can be further expressed as:

$$\begin{aligned} & P(C(t) \leq x) \\ = & P(\Upsilon(t) \leq x) - \frac{\partial}{\partial x} \int_0^t P(\Upsilon(t-u) \leq x)(1 - \frac{\lambda}{r}) du \\ = & P(A_t^N \leq x + rt) - (1 - \frac{\lambda}{r}) \int_0^t \frac{\partial}{\partial x} P(\Upsilon(u) \leq x) du \\ = & \int_0^{x+rt} d(F_{A_t}(y)) - (1 - \frac{\lambda}{r}) \int_0^t \frac{d}{dx} F_{A_u}(ru + x) du, \end{aligned}$$

where, $F_{A_t}(x)$ is the CDF of the input process A_t^N and if we let $f_{A_t}(x)$ denote the *pdf* of A_t^N, we have

$$P(C(t) \leq x) = \int_0^{x+rt} f_{A_t}(y) dy - (1 - \frac{\lambda}{r}) \int_0^t f_{A_u}(ru + x) du. \tag{3.109}$$

The CCDF of $C(t)$ is: $P(C(t) \geq x) = \int_{x+rt}^{\infty} f_{A_t}(y) dy + (1 - \frac{\lambda}{r}) \int_0^t f_{A_u}(ru + x) du$. Applying the

fact that $N\mu\mu_h \to \lambda$ and Eq. (3.43), we have, as $N \to \infty$ and $t \to \infty$:

$$A_t^N \sim N(\lambda t, ct^{2\alpha}), \tag{3.110}$$

where $N(m, \sigma^2)$ is a normal distribution with mean m and variance σ^2. $\alpha = \frac{3-\alpha_{min}}{2}$, and c is the constant in Eq. (3.43).

Then $P(C(t) \geq x)$ can be approximately expressed as:

$$P_r(C(t) \geq x) = \int_{x+rt}^{\infty} \frac{1}{\sqrt{2\pi ct}^{\alpha}} e^{-\frac{(u-\lambda t)^2}{2ct^{2\alpha}}} du + (1 - \frac{\lambda}{r}) \int_0^t \frac{1}{\sqrt{2\pi cu}^{\alpha}} e^{-\frac{(ru-\lambda u+x)^2}{2cu^{2\alpha}}} du, \tag{3.111}$$

where $\alpha = \frac{3-\alpha_B}{2}$, and α_B and c are defined in Eq. (3.43).

We use P_1 and P_2 to denote the two items in Eq. (3.111). Obvious, as $t \to \infty$, $P_1 \to 0$, and the tail distribution of the queue content is determined by P_2.

Although no closed form solution can be obtain, we can calculate numerical results from Eq. (3.111). We set $\lambda = 100kpbs$, $r = 150kpbs$, $c = 1$, and $\alpha = 1.3$, and vary x from 10 to 10000. Fig. 3.17 shows the tail distribution of the content process with the input process being the process generated by the hierarchical model and a Poisson process with the same value of λ. As shown in Fig. 3.17, the traffic generated by the hierarchical model gives rise to a much heavier tail than the traffic governed by the Poisson process.

3.5 Summary

In this chapter we have presented a hierarchical model that has an one-on-one correspondence to protocols in the protocol hierarchy of IP networks and can give insights on how, and to what extent, the user/protocol behavior in each protocol layer contributes to LRD and multifractality. We validate the hierarchical model with real-life traces from IRCache and from Lucent Technologies Bell Labs. We prove, with rigorous derivation, that this simple model can explain both long range dependence at large time scales and multifractality at small time scales. We also analyze the queuing behavior of a fluid queuing system with this hierarchical model as input, and prove that multifractal traffic has the potential of causing a much heavier content tail in a SSQ than a single

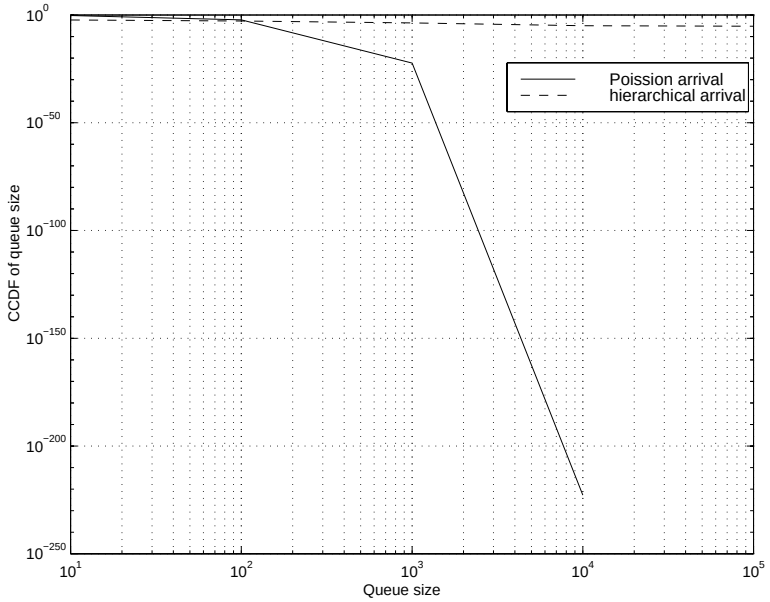


Figure 3.17: Comparison of the CCDF of the queue length with traffic generated by the Poisson process and the hierarchical model as input.

one-tire ON/OFF input.

Chapter 4

Predictive Active Queue Management(PAQM)

In this chapter, we introduce Predictive Active Queue Management (PAQM) in details. Through analytical reasoning, we show that PAQM is a generalized version of RED that takes the future arrival rate as a new dimension of congestion index. By stabilizing the queue at a desirable level with consideration of future traffic, PAQM enables the link capacity to be fully utilized, while not incurring excessive packet loss ratio. Through *ns-2* simulation, we compare PAQM against existing AQM schemes with respect to different performance criteria, and show that under most cases PAQM outperforms SRED in stabilizing the instantaneous queue length, and adaptive virtual queue (AVQ) in reducing packet loss ratio and better utilizing the link capacity.

The rest of the chapter is organized as follows. In Section 4.1, we give an outline of AQM with traffic prediction. Then we delve into the detailed descriptions of PAQM. In particular, in Section 4.2, we elaborate on the controller and derive the expression of packet dropping probability to be used in the next time interval. Following that, we present simulation results in Section 4.3. Finally we conclude the chapter in Section 4.4.

4.1 AQM with Traffic Prediction

Central to the notion of AQM with traffic prediction is prediction of the future traffic based on recent traffic measurements and use of the prediction result to modulate the magnitude of the packet dropping probability at a router. The block diagram is shown in Fig. 4.1. One of the major components in the diagram—the *LMMSE* predictor is depicted in details in Section 2.2.

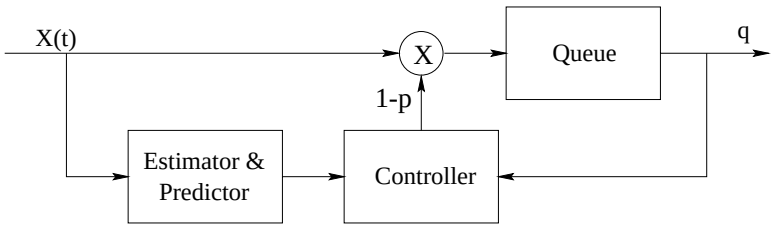


Figure 4.1: AQM with traffic prediction.

Based on the aggregate series samples in Eq. (2.2), the predictor predicts the amount of traffic, $\widehat{X}(n+1) \stackrel{\triangle}{=} X^m(n+1) \cdot m$, and $\widehat{X}(n+2)$, in the next two intervals.

As mentioned in Section 2.2, to implement the *LMMSE* predictor, we have to determine how far ahead traffic prediction is made. This translates into the problem of determining an appropriate value, $\hat{\tau}$, for the interval between two calculation of $X^m(k)$. Fortunately the LRD characteristic of the network traffic implies the relatively low decay of the autocorrelation function, and hence the value of $\hat{\tau}$ is not very critical to the performance. In the simulation study, we set the value of $\hat{\tau}$ to be in the range of 0.02 to 0.05 seconds.

The controller then uses the predicted results to modulate the packet dropping probability, p, to be used in the next time interval. In addition to the objectives of reducing packet loss ratio and improving link utilization, we also aim to keep the queue length at a router at a desirable and stable level.

4.2 Design of the Controller

Recall that in Fig. 4.1 the controller utilizes the prediction results (i.e., the amount of traffic that arrives in the next two intervals) to determine the magnitude of the packet dropping probability. The controller in Fig. 4.1 is not easy to analyze, as a multiplication is involved. Hence we replace the controller with an alternative one in Fig. 4.2. In this system, the output, $d_u(t)$, of the controller is the amount of packets that should be dropped ($0 \leq d_u(t) \leq X(t)$, where $X(t)$ is the amount of traffic (in bytes) that arrive between two packet arrivals). By setting $p = \frac{d_u(t)}{X(t)}$, the two systems are equivalent.

Recall that traffic predication is made on a per-interval basis (where the interval is of duration

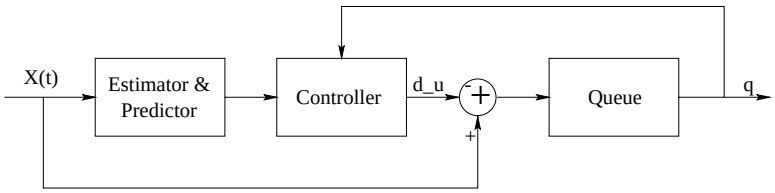


Figure 4.2: An alternate block diagram for AQM with traffic prediction.

$\hat{\tau}$). Let $Q_u(k)$ denote the queue length at the end of the kth interval, and m the number of packets that arrive in an interval of duration $\hat{\tau}$ (which may vary from interval to interval). Then, the discrete-time function of the queue length is

$$Q_u(k+1) = (Q_u(k) + X(k+1) \cdot m - d_u(k+1) - C)^+, \tag{4.1}$$

where $C = R \times \hat{\tau}$ is the number of packets transmitted on the outgoing link and R is the capacity of the outgoing link. The + sign in Eq.(4.1) indicates that the queue length cannot be negative. Since if the queue size $Q_u(k+1) \leq 0$, $d_u(k+1)$ will be automatically set to 0 (no packet can be dropped), in what follows, we only need to calculate $d_u(k+1)$ when $Q_u(k+1) > 0$ without paying much attention to the + sign.

The objective function used is to keep the queue length at an appropriate level or follow an pre-determined, time-variant trajectory. Specifically, let $\widehat{Q}(k)$ and $Q_{opt}(k)$ denote, respectively, the discrete-time functions of the predicted and desirable queue lengths at the end of the kth interval. Then, the objective function we consider is either

$$J = \sum_{i=1}^{M} \left(\widehat{Q}(k+i) - Q_{opt}(k+i) \right)^2, \tag{4.2}$$

or

$$J = \sum_{i=1}^{M} \left(\widehat{Q}(k+i) - Q_{opt}(k+i) \right)^2 + \sum_{i=0}^{M-1} b_i d_u^2(k+i), \tag{4.3}$$

where M is the number of time intervals to be predicted ahead and b_i is the weight for the controller output. Note that the second term in Eq. (4.3) is used to prevent the system from dropping packets excessively. The optimization problem can then be formally stated as

Problem 1: Find $d_u^* = \arg\min_{d_u} J$, where J is expressed in either Eq. (4.2) or Eq. (4.3), subject to $0 \leq d_u(k+i) \leq \widehat{X}(k+i), 1 \leq i \leq M$.

Problem 1 is a constrained optimization problem with concave (quadratic) objective function. Due to the quadratic nature of the objective function, we can first solve it as a unconstrained problem. After that, if the obtained optimal solution falls in the feasible region, then we are done. Otherwise we need to inspect the edges of the feasible region locate the optimal solution. Details will be provided in the following item (**C1**) to (**C4**).

We first solve Problem 1 with the objective function of Eq. (4.2) in the case of $M = 2$ and $Q_{opt}(k) = \overline{Q}, \forall k$, where $\overline{Q}$ is a pre-determined value. To obtain $d_u^*(k)$ and $d_u^*(k+1)$, we take the derivatives of J with respect to $d_u(k+1)$ and with respect to $d_u(k+2)$, and set them to zero:

$$\frac{\partial J}{\partial d_u(k+1)} = 0, \quad \text{and} \quad \frac{\partial J}{\partial d_u(k+2)} = 0.$$

The results of the above equations are (after a few algebraic operations) are

$$\begin{cases} d_u^*(k+1) = Q_u(k) + \widehat{X}(k+1) - C - \overline{Q}, \\ d_u^*(k+2) = \widehat{X}(k+2) - C. \end{cases} \tag{4.4}$$

Using a similar procedure, we can solve problem 1 with the objective function of Eq. (4.3) in the case of $M = 2$ and $Q_{opt}(k) = \overline{Q}, \forall k$. The results are

$$\begin{cases} d_u^*(k+1) = -\frac{(4b_1-1)(Q_u(k)+\widehat{X}(k+1)-\overline{Q})+2b_1\widehat{X}(k+2)+(1-6b_1)C}{1-4b_1-2b_0+4b_0b_1}, \\ d_u^*(k+2) = -\frac{2b_0(Q_u(k)+\widehat{X}(k+1)-\overline{Q})+(2b_0-1)\widehat{X}(k+2)+(1-4b_0)C}{1-4b_1-2b_0+4b_0b_1}. \end{cases} \tag{4.5}$$

Note that Eq. (4.5) reduces to Eq. (4.4) when $b_1 = b_2 = 0$. As $d_u^*(k+1)$ and $d_u^*(k+2)$ obtained in Eq. (4.5) may violate the constraints of $0 \leq d_u(k+1) \leq \widehat{X}(k+1)$ and $0 \leq d_u(k+2) \leq \widehat{X}(k+2)$, we consider the following four cases:

(C1) If $d_u^*(k+1) \in [0, \widehat{X}(k+1)]$ and $d_u^*(k+2) \in [0, \widehat{X}(k+2)]$, then $d_u(k+1) \leftarrow d_u^*(k+1)$ and $d_u(k+2) \leftarrow d_u^*(k+2)$.

(C2) If $d_u^*(k+1) \notin [0, \widehat{X}(k+1)]$ and $d_u^*(k+2) \notin [0, \widehat{X}(k+2)]$, then the optimal solution is one of

the four possible pairs, $(0,0)$, $(\widehat{X}(k+1),0)$, $(0,\widehat{X}(k+2))$ and $(\widehat{X}(k+1),\widehat{X}(k+2))$. One can compute $J(0,0)$, $J(\widehat{X}(k+1),0)$, $J(0,\widehat{X}(k+2))$ and $J(\widehat{X}(k+1),\widehat{X}(k+2))$ and select the one that gives the smallest value.

(C3) If $d_u^*(k+1) \in [0,\widehat{X}(k+1)]$ and $d_u^*(k+2) \notin [0,\widehat{X}(k+2)]$, then the optimal solution is one of the two possible pairs, $(d_u^*(k+1),0)$ and $(d_u^*(k+1),\widehat{X}(k+2))$. One can compute $J(d_u^*(k+1),0)$ and $J(d_u^*(k+1),\widehat{X}(k+2))$ and determine the optimal controller output accordingly.

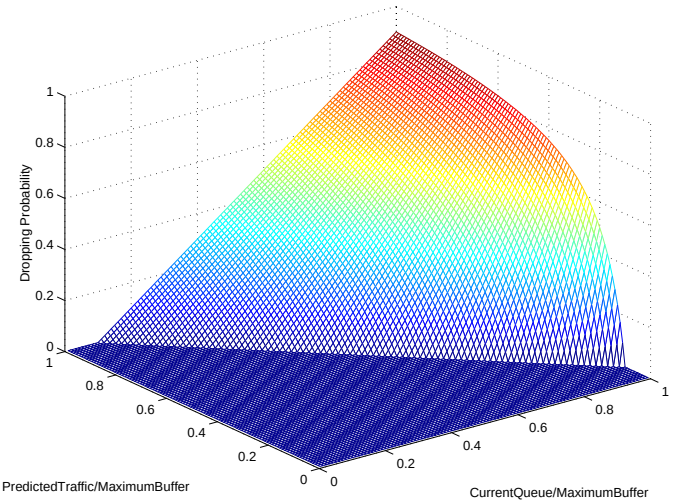
(C4) If $d_u^*(k+1) \notin [0,\widehat{X}(k+1)]$ and $d_u^*(k+2) \in [0,\widehat{X}(k+2)]$, then the optimal solution is one of the two possible pairs, $(0,d_u^*(k+2))$ and $(\widehat{X}(k+1),d_u^*(k+2))$. One can compare $J(0,d_u^*(k+2))$ and J$(\widehat{X}(k+1),d_u^*(k+1))$ and determine the optimal controller output accordingly.

After $d_u(k+1)$ is determined, we set $p(k+1) = \frac{d_u(k+1)}{\widehat{X}(k+1)}$ as the packet dropping probability to be used in the next interval. For example, if we use d_u^* derived in Eq. (4.4), then

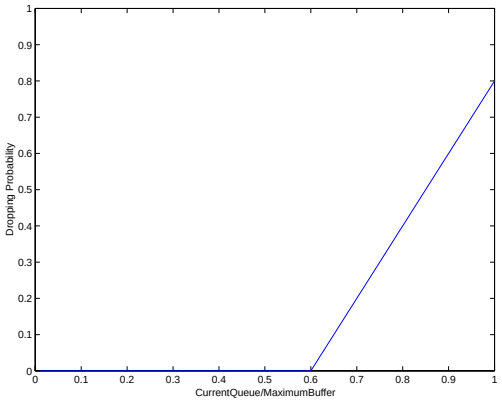
$$p(k+1) = \begin{cases} 0, & Q_u(k) < C + \overline{Q} - \widehat{X}(k+1), \\ \frac{Q_u(k)+\widehat{X}(k+1)-C-\overline{Q}}{\widehat{X}(k+1)}, & C + \overline{Q} - \widehat{X}(k+1) < Q_u(k) < C + \overline{Q}, \\ 1, & Q_u(k) > C + \overline{Q}. \end{cases} \tag{4.6}$$

Note that $p(k+1)$ is a linear function of $Q_u(k)$ when $\widehat{X}(k+1)$ is at a fixed value.

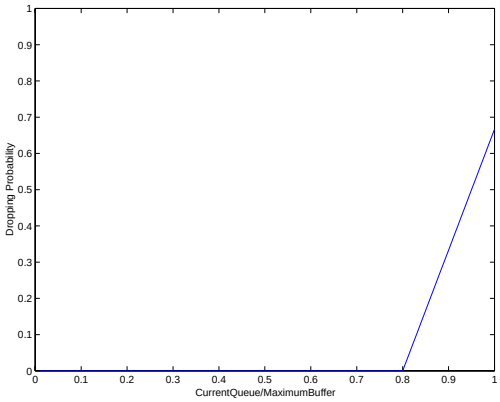
Fig. 4.3 gives the values of $p(k+1)$ as a function of $Q_u(k)$ and $\widehat{X}(k+1)$, with $Q_{opt} = 0.6\cdot$ MaximumBufferSize and $R = (0.5\cdot$ MaximumBufferSize $)/\hat{\tau}$ (i.e., half of the queue can be emptied in one measurement interval). As shown in Fig. 4.3 (b)-(c), the projection of $p(k+1)$ onto the plane that corresponds to a fixed value of $\widehat{X}(k+1)$ is a RED-like curve. That is, RED can be viewed as a special case of PAQM with the estimated future incoming traffic fixed at certain value. Another interpretation of Fig. 4.3 is that RED can be stabilized at a desirable queue length, if it considers the amount of incoming traffic and sets the values of the two thresholds, *min_th* and *max_th* in compliance with the curves given in Fig. 4.3. Fig. 4.4 gives the packet dropping probability, $p(k+1)$, calculated in an *ns-2* simulation with the same setting as in Fig. 4.3. Comparing Fig. 4.4 against Fig. 4.3, we observe that the packet dropping probability used by a router in the simulation indeed lies on the plane depicted in Fig. 4.4.



(a) Dropping probability as a function of current queue size and predicted incoming traffic



(b) $\widehat{X}(k+1) = 0.5 \cdot$ MaximumBufferSize



(c) $\widehat{X}(k+1) = 0.3 \cdot$ MaximumBufferSize

Figure 4.3: The packet dropping probability, $p(k+1)$, to be used in the next time interval versus the queue length and the estimated traffic. The values of $Q_u(k)$ and $\widehat{X}(k+1)$ are normalized with respect to the maximum buffer size.

Discussion on the stationarity of the Internet traffic The usage of *LMMSE* predictor requires the wide sense stationarity (WSS) of the Internet traffic so that the autocorrelation function $r(t)$ can be defined. But in reality, the traffic going through a router presents a cyclic behavior with highest volume reached during diurnal hours and lowest volume during night hours. This may raise a hindrance in applying *LMMSE* predictor. But, after looking into the Internet traffic in smaller time scales, it is still promising in applying *LMMSE* predictor based on the notion of short time wide sense stationarity (STWSS), i.e., for a short time period (in the scale of minutes or hours), the Internet traffic can be viewed as WSS. To this end, in order to accommodate the cyclic

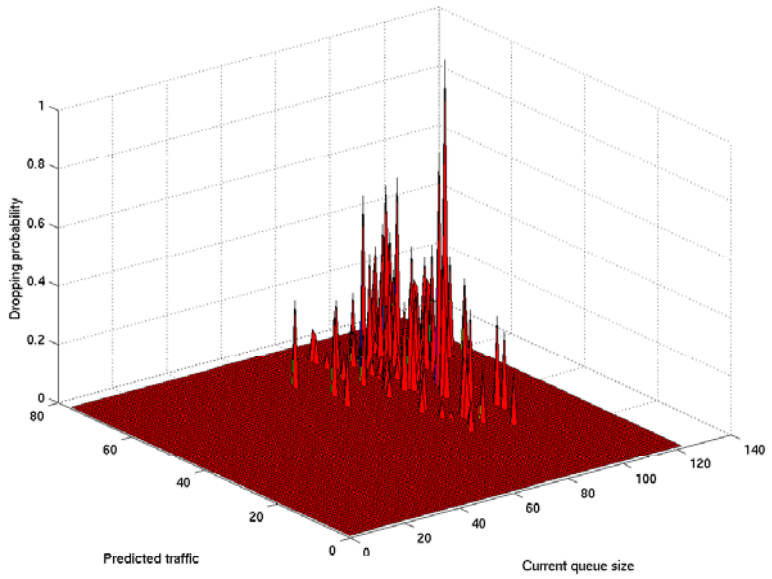


Figure 4.4: The packet dropping probability, $p(k+1)$, calculated in an *ns-2* simulation run.

behavior of the traffic, the optimal queue size (Q_{opt}) should be adjusted adaptively (take larger values for diurnal hours and smaller values for night hours). Therefore, for short time periods, the *LMMSE* predictor can still be used to capture the trend of the incoming traffic and to modulate the packet dropping probability. This also strengthens the importance of keeping $\hat{\tau}$ small (0.02 to 0.05 second), since if $\hat{\tau}$ is too large, the *LMMSE* predictor may not generate correct index on the trend of the forthcoming traffic (due to the long-term un-stationarity of the Internet traffic) and hence the goal of $PAQM$ cannot be achieved.

Discussion on the open/close-looped control and prediction Recall that in the PAQM scheme, we determine the amount of traffic to be dropped, $d_u(k)$, so that the queue length at a router can be stabilized. The larger the $d_u(k)$, the more connections will experience packet losses and reduce their sending rates (by shrinking their TCP congestion windows). After a period of about one RTT, the incoming traffic $X(k+1)$ decreases and causes $d_u(k)$ decrease accordingly.

If we only consider the control process at the queue (see Fig. 4.2) the system is close-looped. But, in a bird's-eye of the entire system composed of the AQM controller at the router and the TCP congestion control system at the end hosts, the PAQM control scheme is open-looped. Since in PAQM, we didn't take into consideration the TCP's behavior, the effect of dropping probability

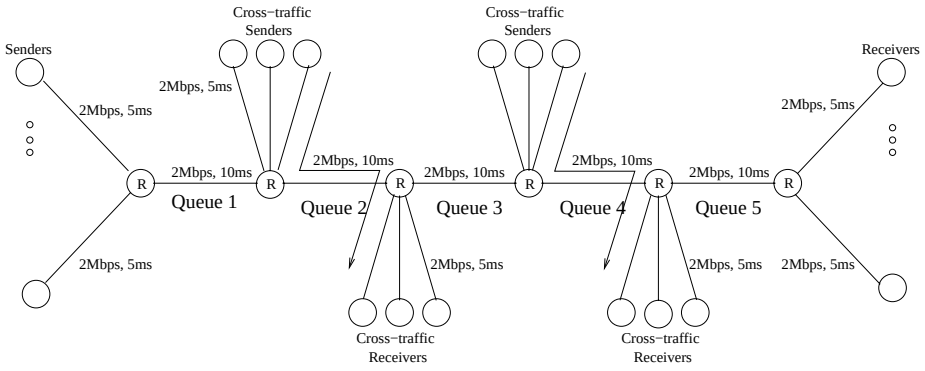


Figure 4.5: The multiple bottleneck simulation topology.

on TCPs' sending rate, and the delay between the instance when a packet is dropped and when the incoming traffic starts to decrease. The reason why we adopt an open-looped prediction and control system is that a close-looped counterpart needs extremely accurate and detailed information about the network, such as network topology, accurate TCP model, number of TCP connections and their RTTs, instantaneous congestion window sizes, etc.. Furthermore, obtaining such information is not a trivial task in current best-effort networks due to the inherent bursty characteristics of the Internet traffic. Even if we can collect all the information, there are still two challenges. First, how to systematically integrate all the information into a close-looped system is a huge hindrance we need to hurdle. Second, how to make the close-looped system robust under network conditions (topology, parameters, etc.) changes. As we know the gap between the time instance when the changes happen and when we can detect these changes exists and cannot be easily shortened.

Therefore, we adopt an open-looped prediction and control system which is more feasible and robust than a close-looped on. As we have introduced, we capture the dynamics of the Internet traffic by traffic prediction and all the operations depend only on the prediction results. As long as the prediction is accurate (as shown in Chapter 2) our objective in stabilizing the queue length can be faithfully achieved.

4.3 Simulation Results

We have implemented PAQM (with the *LMMSE*, *simple*, or *trivial* traffic predictors) along with RED [53], SRED [98], and AVQ [77] in ns-2, and conducted a simulation study to validate the

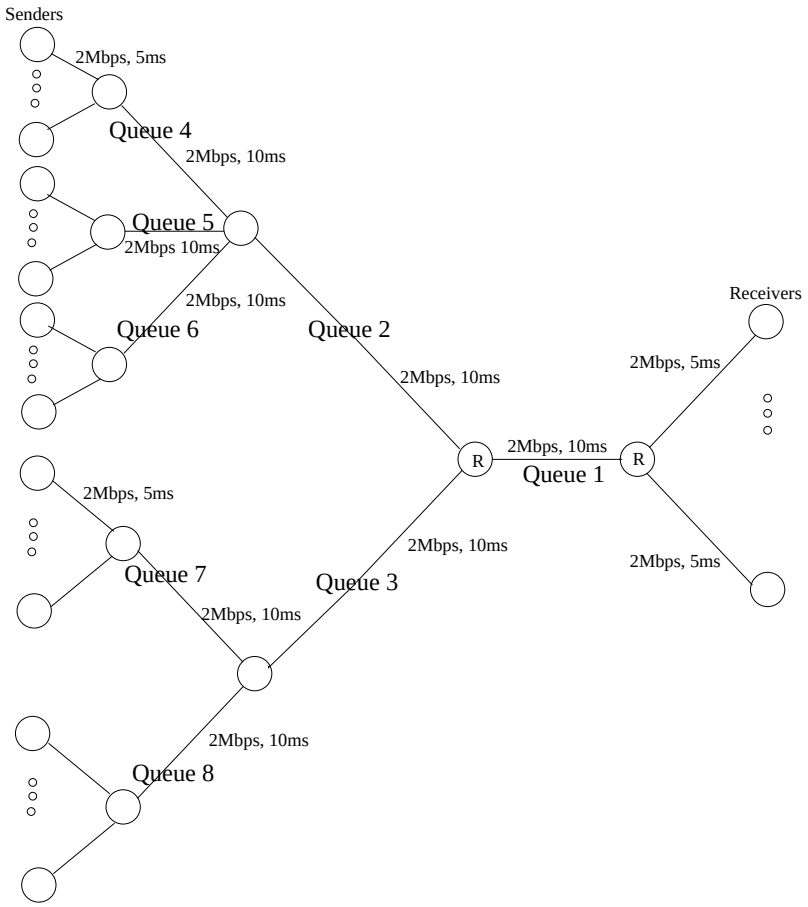
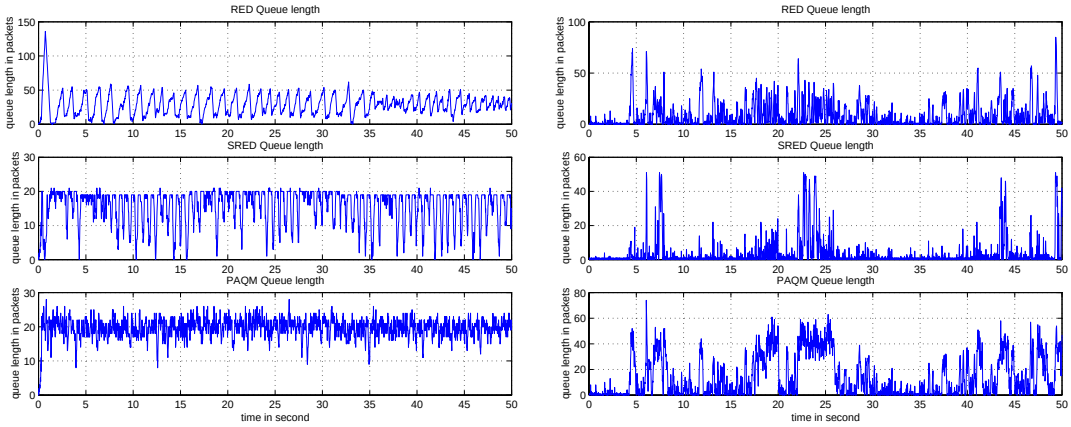


Figure 4.6: The arbitrary simulation topology.

proposed design and compare the performance of PAQM against the other schemes. We examine the behavior of these schemes under a variety of network topologies and traffic sources. In particular, we have considered the network topologies with a single bottleneck link, with multiple bottleneck links (e.g., Fig. 4.5), as well as of arbitrary topologies (e.g., Fig. 4.6). The maximum buffer size of each router is set to 100 packets (of size 1000 bytes) under PAQM, RED, and AVQ, and 20 packets under SRED. (The reason for choosing a different maximum buffer size under SRED is that through extensive simulation, we find that SRED always attempts to keep the queue close to full. The value is so chosen that the queue length under SRED is comparable to that under PAQM or RED.) We use an assortment of traffic sources (namely TCP sources that generate packets according to the on-off model or real traffic traces down-loaded from the Internet, and constant bit rate UDP sources).



(a) on-off model (b) real traffic trace

Figure 4.7: Instantaneous queue length in the single bottleneck network with TCP sources.

The parameters used in PAQM are: Q_{opt} is set to 20 packets[1], and $\hat{\tau}$ is in the range of 0.02-0.05 seconds. The parameters of RED and SRED are set, respectively, in compliance with the guidelines available in [54] and [98] (i.e., for SRED $M = 1000$, $\alpha = 1/M = 0.001$, and $p_{max} = 0.15$). Finally, the desirable utilization, γ, of AVQ is set to 0.98, and the damping factor, α, is determined in compliance with Theorem 1 in [77] to ensure system stability ($\alpha = 0.15$).

Each data point is the result averaged over 10 simulation runs. We report on a small set of the simulations which we believe is the most representative. In spite of numerous system parameters involved, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves for a representative set of parameter values (reported below) is valid over a wide range of parameter values.

4.3.1 Comparison between RED, SRED, and PAQM

In this set of experiments we compare PAQM against RED and SRED with respect to instantaneous queue length, packet loss ratio, and total throughput attained by receivers under different network topologies and traffic mixes. The *LMMSE* predictor is used to infer the amount of traffic that will arrive in the next two measurement intervals.

Simulation results under the single bottleneck topology k TCP connections are established over a single bottleneck link of capacity 2 Mbps, where k varies from 20 to 100. Fig. 4.7 gives

[1] As will be shown later, the performance of PAQM is not very sensitive to Q_{opt}.

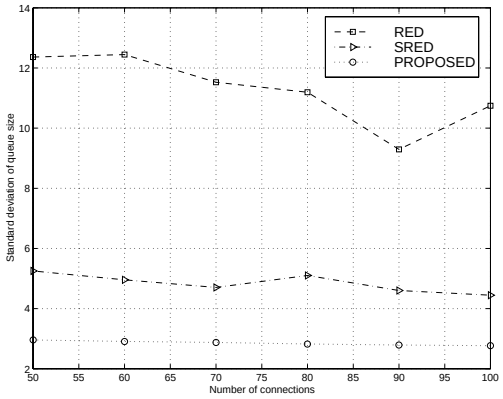
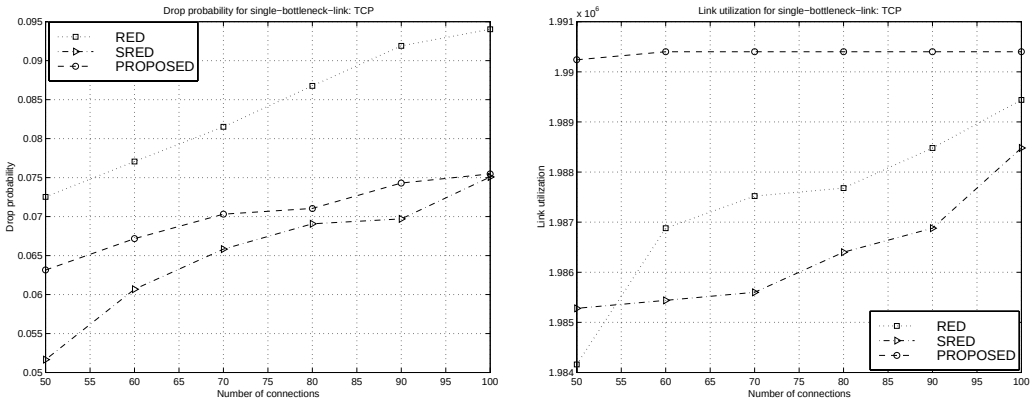


Figure 4.8: The standard deviation of the instantaneous queue length in the single bottleneck network.



(a) packet loss ratio (b) attainable throughput

Figure 4.9: Comparison of RED, SRED, and PAQM with respect to the packet loss ratio and attainable throughput in the single bottleneck link network.

the instantaneous queue length in the cases that 60 TCP connections are established and generate packets using either the on-off model ((a)) or real traffic traces ((b)). Fig. 4.8 gives the standard deviation of instantaneous queue length under the same setting as in Fig. 4.7 (a) (except that the number of connections varies from 50 to 100). As compared to RED and SRED, PAQM indeed better stabilizes the queue at the desirable level.

Fig 4.9 gives the packet loss ratio and the throughput attained by all receivers under the case of TCP sources with the on-off model. PAQM performs better than SRED w. r. t. attainable throughput, but slightly worse than SRED w. r. t. packet loss ratio. This is because SRED always attempts to keep the queue (close to) full, and hence does not pro-actively drop packets

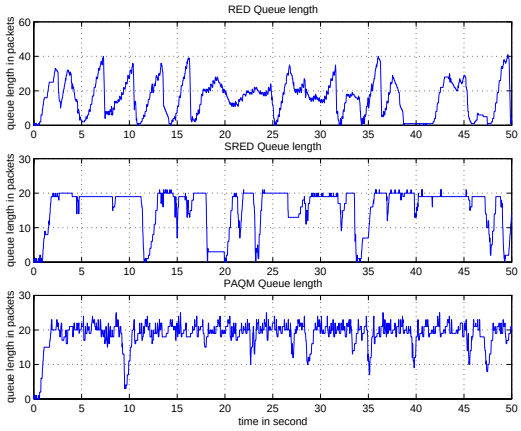


Figure 4.10: The instantaneous queue lengths at queue 2 in the multiple bottleneck network.

that aggressively. However, keeping the queue full also makes the queue more susceptible to the global synchronization effect. This is evidenced in Fig. 4.7 (a) that the instantaneous queue length frequently oscillates between empty and full. This also accounts for the fact that SRED does not attain as much throughput as PAQM.

Simulation results under the multiple bottleneck network We repeat in a network with multiple bottlenecks the same experiments that we carried out in the single bottleneck network. As shown in Fig. 4.5, there are 5 queues along the end-to-end path, among which queue 2 and queue 4 are shared by cross traffic. Altogether k TCP connections are established with senders (receivers) at the left (right) hand side, where k varies from 20 to 100. Each cross traffic bundle is composed of $0.5k$ TCP connections. In our extensive simulation study, the queue length at queue 5 is always 0 or 1, indicating that the link is not a bottleneck link. The other four queues exhibit similar trends as far as the performance comparison is concerned, and hence, we only depict the instantaneous queue length, the packet loss ratio and attainable throughput of queue 2 in Figs. 4.10 and 4.12, respectively. PAQM outperforms the other two schemes with respect to all three measures. The reason why SRED does not perform as well in terms of packet loss ratio is perhaps because as SRED has the tendency to keep the queue (close to) full, packet losses are more likely to occur as a result of buffer overflow, when the link becomes more congested with cross traffic. We have also conducted simulation on several networks of arbitrary topologies and got similar results.

Simulation results in the case of mixed traffic sources As reported in [120], TCP traffic constitutes the majority ($\geq 85\%$) of the Internet traffic. To evaluate the performance of PAQM

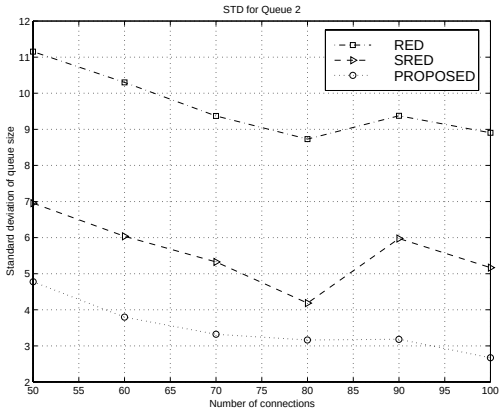
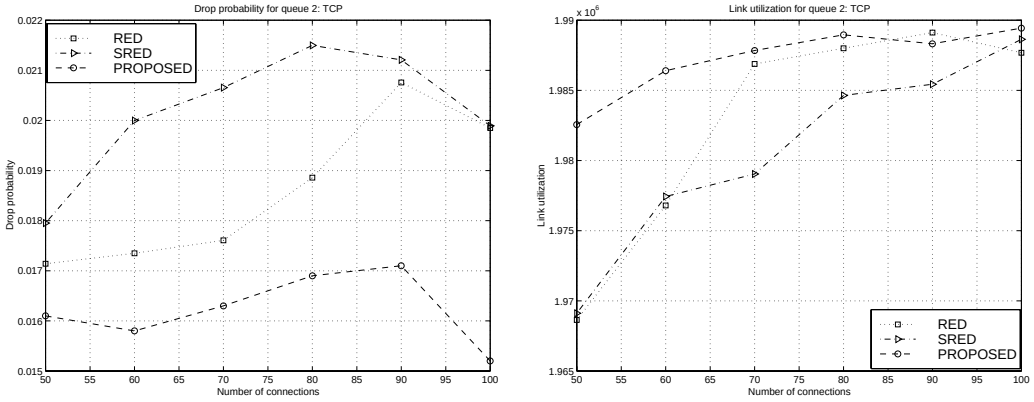


Figure 4.11: The standard deviation of the instantaneous queue length at queue 2 in the multiple bottleneck network.



(a) loss ratio (b) attainable throughput

Figure 4.12: Comparison of RED, SRED, and PAQM with respect to the packet loss ratio and attainable throughput at queue 2 in the multiple bottleneck network.

under a more realistic traffic mix, we repeat the experiments with 85% of the connections being TCP and 15% being UDP in the multiple-bottleneck network. All the parameters used in RED, SRED and PAQM remain the same, except that we have to change the maximum buffer size under SRED to 30 packets so that comparisons can be fairly made. This is again due to the fact that SRED tends to keep the queue (close to) full especially under heavy traffic load.

Figs. 4.13 and 4.14 give the standard deviation of instantaneous queue length, packet loss ratio, and attainable throughput, respectively. SRED achieves slightly better performance in terms of stabilizing the instantaneous queue length, but this is really achieved by keeping the queue (close to) full. As UDP traffic is non-responsive to packet losses and persist in sending packets, there will

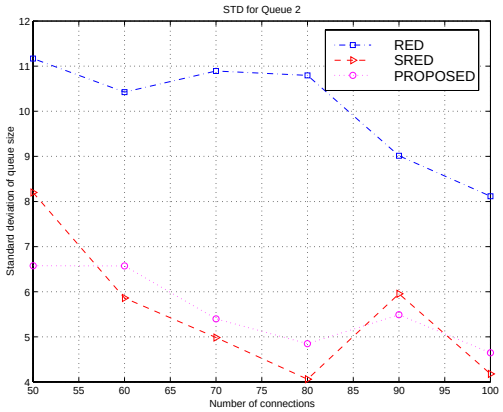
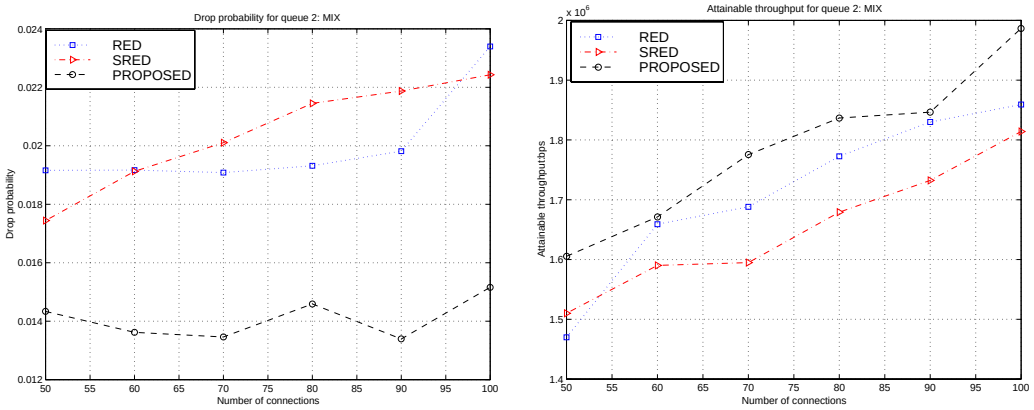


Figure 4.13: The standard deviation of the instantaneous queue length at queues 2 in the case of mixed traffic sources.



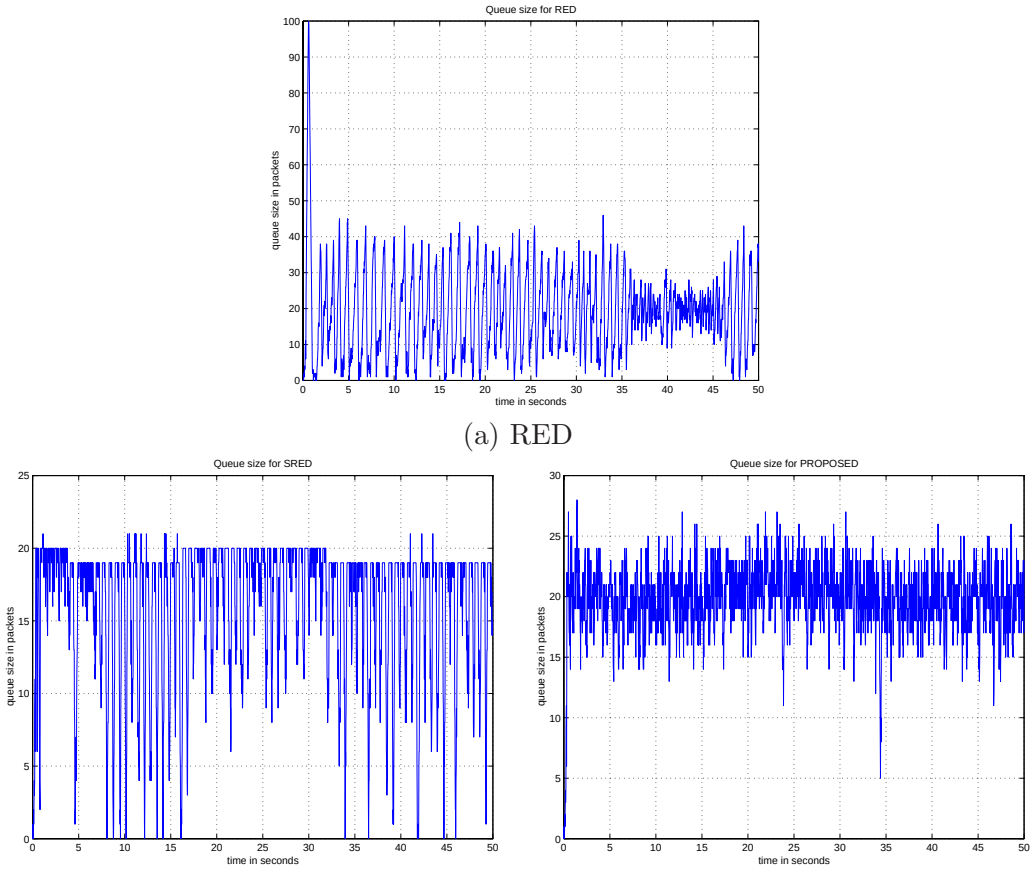
(a) loss ratio (b) attainable throughput

Figure 4.14: Comparison of RED, SRED, and PAQM with respect to the packet loss ratio and attainable throughput at queue 2 in the case of mixed traffic sources.

not be a sustained period of low link utilization following packet losses. That is why SRED always maintains a stable (and full) queue. The consequent effect is that it also incurs larger packet loss ratio and attains less throughput as compared to PAQM.

Simulation results in the case of dynamic connection establishment and termination

To test the responsiveness in stabilizing the queue in the case that connections are dynamically established and terminated, we repeat the same experiments in the single bottleneck network except that initially 40 TCP connections are established, at time 10 another 20 connections are established, and finally at time 30, 20 connections are terminated. All the simulation runs last for 50 seconds. Fig. 4.15 gives the instantaneous queue length under RED, SRED, and PAQM. The queue length



(a) RED

(b) SRED

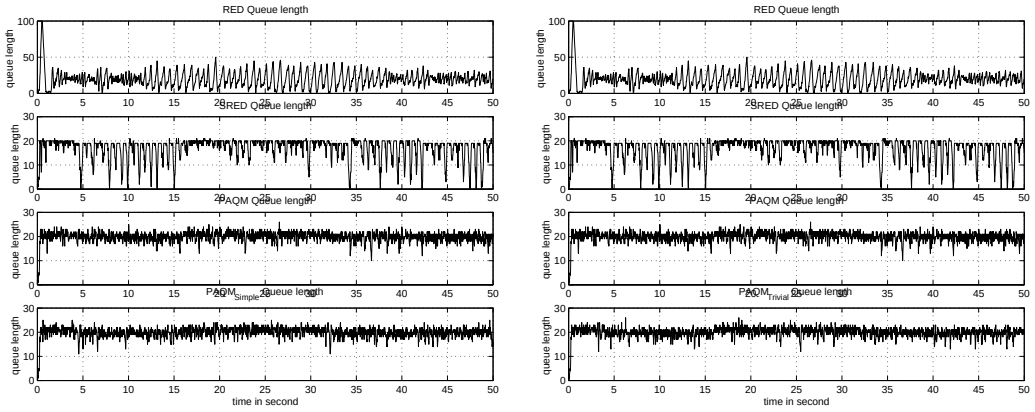
(c) PAQM

Figure 4.15: The instantaneous queue length in the case of dynamic connection establishment and termination.

under PAQM is always stable around $Q_{opt} = 20$ packets, regardless of change of the number of connections. The queue size under RED, on the other hand, oscillate dramatically almost all the times. SRED again keeps its queue length close to the maximum buffer size, and hence is subject to the global synchronization effect experienced by a drop-tail queue. This is evidenced by the fact that the instantaneous queue length frequently oscillates between empty and full.

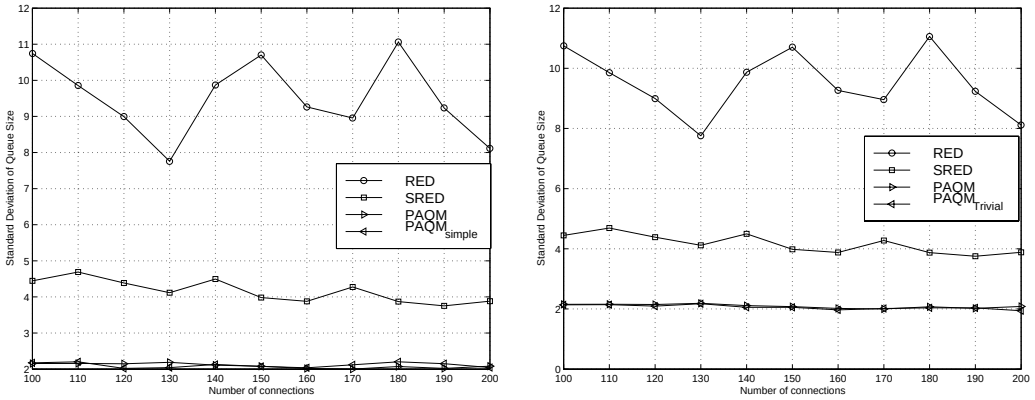
4.3.2 Comparison between *LMMSE*, *Simple*, and *Trivial*

We repeat the same experiments in the single bottleneck network, but use the various traffic predictors introduced in Chapter 2 to infer the amount of future traffic. Figs. 4.16–4.17 give, respectively, the instantaneous queue length and its standard deviation in the case that 100 TCP connections are established and generate packets using the on-off model. (Note that we use PAQM, PAQM$_{simple}$



(a) *simple* predictor (b) *trivial* predictor

Figure 4.16: Instantaneous queue length under RED, SRED, PAQM, PAQM$_{simple}$ and PAQM$_{trivial}$.

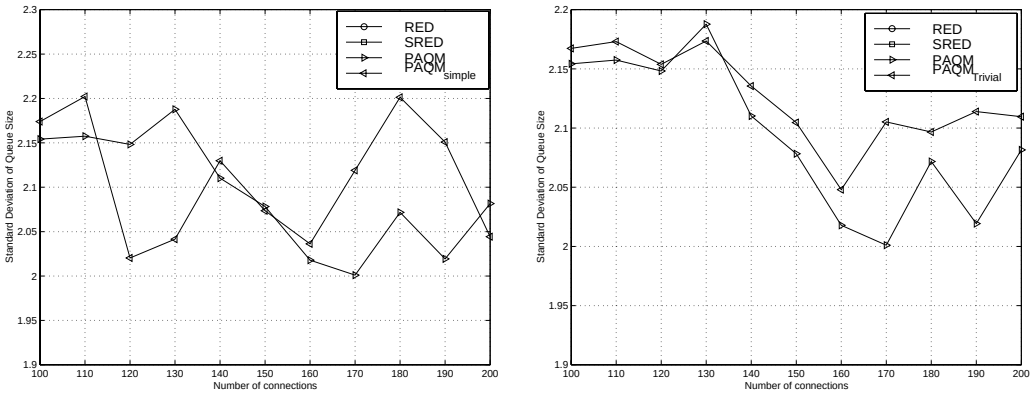


(a) *simple* predictor (b) *trivial* predictor

Figure 4.17: The standard deviation of the instantaneous queue length under RED, SRED, PAQM, PAQM$_{simple}$ and PAQM$_{trivial}$.

and PAQM$_{trivial}$ to denote, respectively, the PAQM scheme equipped with the *LMMSE* predictor, the *simple* predictor, and the *trivial* predictor.) As shown in Figs. 4.16–4.17, both PAQM$_{simple}$ and PAQM$_{trivial}$ achieve similar performance to PAQM and can keep the queue length at around 20 packets. The standard deviations of the instantaneous queue length under PAQM, PAQM$_{simple}$ and PAQM$_{trivial}$ are very close to each other, and are much smaller than those under SRED and RED.

In order to further differentiate the performance of PAQM, PAQM$_{simple}$, and PAQM$_{trivial}$, we depict in Fig. 4.18 the "enlarged" version of Fig. 4.17. As shown in Fig. 4.18 (a), when the number of connections is less than 150, PAQM$_{simple}$ performs slightly better than $PAQM$, and vice versa.



(a) PAQM vs PAQM$_{simple}$ (b) PAQM vs PAQM$_{trivial}$

Figure 4.18: The standard deviation of the instantaneous queue length under PAQM, PAQM$_{simple}$ and PAQM$_{trivial}$.

As shown in Fig. 4.18 (b), the queue length under PAQM is almost always more stable than that under PAQM$_{trivial}$. These results are consistent with those presented in Section 2.3, where we find that the *LMMSE* and *simple* predictors outperform the *trivial* predictor in terms of prediction accuracy. Let the prediction error be $\sigma \stackrel{\triangle}{=} X(k+1) - \widehat{X}(k+1)$. By plugging the expression of σ into the first equation of Eq. (4.4), we have

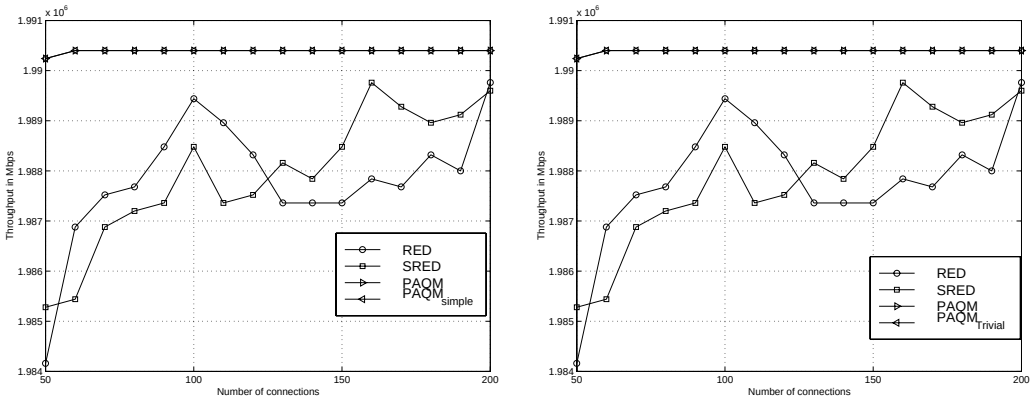
$$d_u^*(k+1) = Q_u(k) + X(k+1) - \sigma - C - \overline{Q}. \tag{4.7}$$

Plugging Eq. (4.7) in $Q_u(k+1) = Q_u(k) + X(k+1) - d_u^*(k+1) - C$, we have $Q_u(k+1) = \overline{Q} + \sigma$. Hence, if the smaller the prediction error, the more stable the queue length.

Figs. 4.19–4.20 depict, respectively, the attainable throughput and the packet loss ratio under PAQM, PAQM$_{simple}$, and PAQM$_{trivial}$. Similar conclusions can be made on the performance comparison with respect to these two measures between PAQM, PAQM$_{simple}$, and PAQM$_{trivial}$.

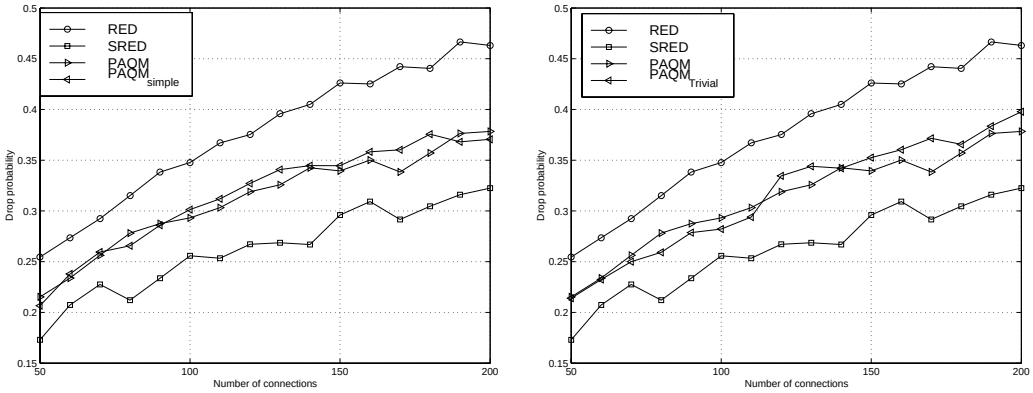
4.3.3 Comparison between AVQ and PAQM

As mentioned in Section 2.6, although PAQM is not targeted to decouple the congestion measure and the performance measure, the fact that it takes into account of the amount of future traffic in the next two measurement intervals (or equivalently, the future arrival rate) does help to achieve this objective. To illustrate this, we repeat the same experiment in the single bottleneck network, but



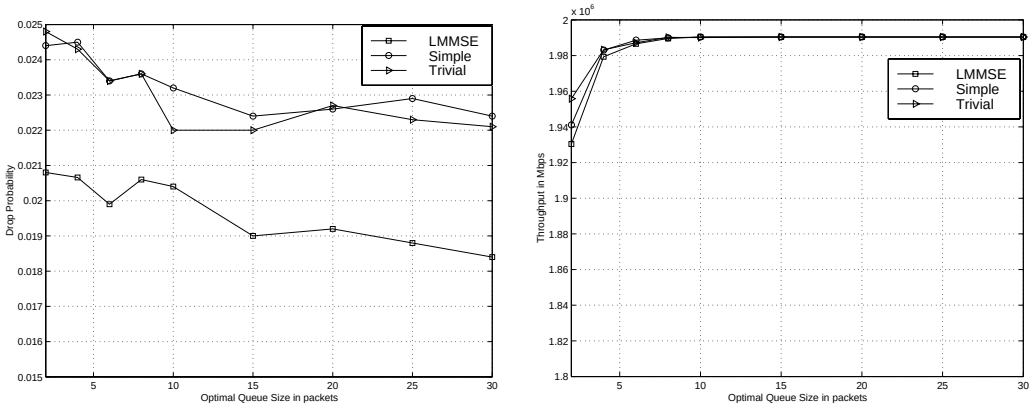
(a) PAQM vs PAQM$_{simple}$ (b) PAQM and PAQM$_{trivial}$

Figure 4.19: The attainable throughput under PAQM, PAQM$_{simple}$ and PAQM$_{trivial}$.



(a) PAQM and PAQM$_{simple}$ (b) PAQM and PAQM$_{trivial}$

Figure 4.20: The packet loss ratio under PAQM, PAQM$_{simple}$ and PAQM$_{trivial}$.

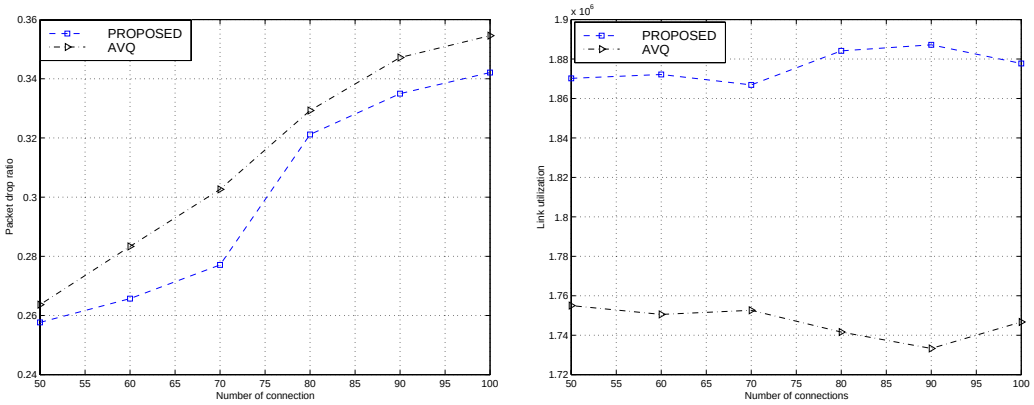


(a) packet loss ratio (b) attainable throughput

Figure 4.21: The packet loss ratio and link utilization versus Q_{opt} under PAQM, PAQM$_{simple}$, and PAQM$_{trivial}$.

change Q_{opt} from 2 to 30, and measure the packets loss ratio and attainable throughput. Fig. 4.21 gives the simulation results with 30 TCP connections going through the bottleneck link (the number of TCP connections k changes from 20 to 100, for the ks we observed similar results). As Q_{opt} changes from 2 to 30, the packet loss ratio changes from 0.0206 to 0.0185 under PAQM (and from 0.025 to 0.022 under *simple* and *trivial*), while the attainable throughput changes from 1.93 Mbps to 1.99 Mbps (the latter levels off when $Q_{opt} \geq 8$). This suggests that under PAQM the equilibrium value of the congestion measure (queue length) is independent of that of the performance measure (packet loss or attainable throughput).

As PAQM does exhibit the decoupling behavior, we compare it against AVQ — the scheme currently reported in [77] to give the best performance in the second category of AQM schemes (Table 2.2). Again we repeat the same experiments in the single bottleneck network except that we set $Q_{opt} = 2.5$ under PAQM. This is because AVQ usually keeps its average queue length at 2.5 packets under the given simulation setting. By setting $Q_{opt} = 2.5$ packets, a fair comparison can then be made with respect to packet loss ratio and attainable throughput. Fig. 4.22 gives the simulation results. The average queue length both under AVQ and PAQM is kept around 2.5 packets, but PAQM achieves better performance both in terms of packet loss ratio and attainable throughput. This demonstrates the powerfulness of exploiting LRD and taking into account of future arrival rate in determining the packet dropping probability. We have observed similar trends in simulation runs conducted in the multiple bottleneck network and in networks of arbitrary



(a) packet loss ratio (b) link utilization

Figure 4.22: Performance comparison between AVQ and PAQM.

topologies.

4.4 Summary

We have explored in this chapter the issue of exploiting traffic predictability to enhance the performance of AQM. We show that the correlation structure present in long-range dependent traffic can be used to accurately predict the future traffic. We then exploit the prediction results in the calculation of the packet dropping probability. By stabilizing the instantaneous queue length at a desirable level (in anticipation of future traffic), PAQM enables the link capacity to be fully utilized, while not incurring excessive packet loss ratio. Through *ns-2* simulation, we show that under most cases PAQM outperforms SRED in stabilizing the instantaneous queue length, and AVQ in reducing packet loss ratio and utilizing the link capacity.

It is worth mentioning that PAQM is orthogonal to REM, PI, and AVQ in the sense that the effect of future incoming traffic (predicted through exploitation of LRD) can be figured in the calculation of the price function or in the adjustment of the virtual queue capacity to further improve their performance. As PAQM has been shown to be a generalized version of RED with future traffic figured in (Fig. 4.3), one can expect that REM/PI/AVQ, when coupled with PAQM, also gives a more generalized version expectedly with better performance.

Chapter 5

TCP with Traffic Prediction

In this chapter, we propose a novel scheme, *TCP with traffic prediction* (or *TCP-TP*). We show that the correlation structure present in LRD traffic can be used to predict the future traffic at least one round-trip time (RTT) ahead. This is realized with the use of a traffic predictor that minimizes the linear mean square errors (*LMMSE*) as introduced in Chapter 2. The prediction results are then used to infer the optimal operational point at which a TCP connection should operate. By optimal operational point, we mean the point that achieves fairness among the connections that traverse the same bottleneck link. Specifically, as shown in Fig. 5.1, the vertical and horizontal axes represent the throughput, Tf, attained by the TCP connection of interest and that, TB, by the background traffic, respectively. Let the number of connections that traverse the bottleneck link be denoted as N and the bandwidth of the bottleneck link as C. Then the optimal operational point is the joint point of line $Tf + TB = C$ (which we call the capacity line) and line $\frac{Tf}{TB} = \frac{1}{N-1}$ (which we call the fairness line). Without traffic prediction, a TCP connection usually reaches the optimal point through several additive increase (AI) and multiplicative decrease (MD) phases (Fig. 5.1). With accurate traffic prediction, we show that if all TCP connections are synchronized in making their congestion control decisions and prediction results are accurate, a TCP connection can reach the optimal point in one RTT without commencing MD phases (and hence without incurring packet losses), This leads to significant performance improvement in terms of fast convergence to the optimal operational point, packet loss ratio, and attainable throughput. The above results are corroborated by *ns-2* simulation and empirical experimentation over the Internet.

In the case of existence of prediction errors, we show via phase plots that with the use of the MD phase, TCP-TP can still retain the stability established in the AIMD algorithm. Moreover,

we analyze rigorously the impact of prediction errors on fairness, and show when the prediction error is 100% (i.e., the predicted value is twice as large as the original value), the index of fairness only degrades 2.5%. Finally we demonstrate the change needed to incorporate the traffic prediction extension into TCP is minimal (tens of lines of code changes) by implementing TCP-TP both in *ns-2* and in FreeBSD 4.1 and conducting experiments. In the case that TCP connections do not synchronize in window adjustment and are subject to different RTTs, we show via *ns-2* simulation and FreeBSD implementation and experimentation that both TCP and TCP-TP cannot achieve fairness, but a TCP-TP connection still performs better than TCP (67% improvement in terms of packet loss ratio).

TCP-TP is especially well-suited for improving the performance of long-lived TCP connections, as short-lived TCP connections may not reach the optimal operational points before they terminate. Although the number of short-lived TCP connections are much larger than that of long-lived TCP connections, as reported in [27], the majority of Internet traffic is still dominated by long-lived TCP connections and long-lived TCP connections play much more important roles in network utilization. As TCP-TP improves the attainable throughput of long-lived TCP connections, it helps to increase the overall network utilization. On the other hand, it has been suggested in the work of *Congestion Manager* (CM) [62] that (short-lived) TCP connections destined for the same destination (e.g., web downloads from a web server) should be bundled together and subject to the same congestion control (so as to avoid blind competition among concurrent connections). TCP-TP can be used in conjunction with CM.

The rest of the chapter is organized as follows: In Section 5.1, we give an overview of TCP-TP. Then, we delve into the technical details of TCP-TP in Section 5.2. In particular, we elaborate on how a TCP-TP sender predicts its future attainable throughput and adjusts its congestion window. In Section 5.3, we analyze how prediction errors impact the performance in terms of fairness. In Section 5.4, we present *ns-2* simulation results and FreeBSD 4.1 implementation based experiments. The chapter concludes with Section 5.5.

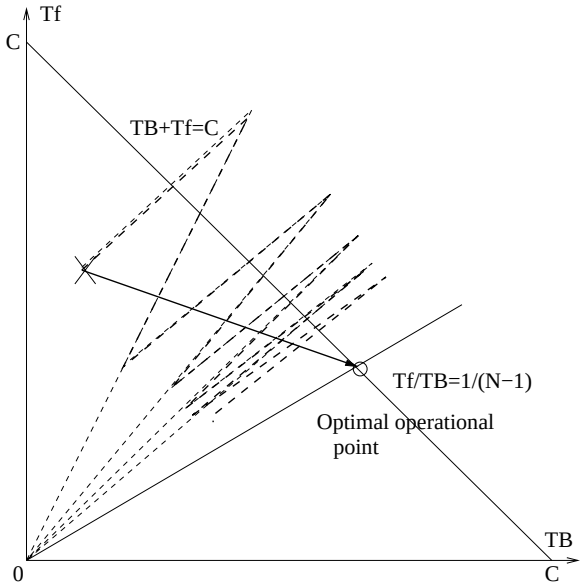


Figure 5.1: The phase plot that illustrates how fairness is achieved in the case that N connections traverse a bottleneck link. A TCP connection usually goes through several AI and MD phases, and follows the dashed line to reach the optimal operational point.

5.1 Exploitation of LRD Characteristics in TCP Congestion Control

5.1.1 An Overview of TCP-TP

The rationale behind exploitation of traffic prediction in TCP congestion control is to enable TCP to predict, with exploitation of the correlation structure across multiple time scales, its attainable throughput at least one round trip time (RTT) ahead. With the predicted information, the TCP connection then infers the optimal operational point, and adjusts the window increase/decrease in its congestion control (in particular, the AI phase of the *AIMD* algorithm). The window adjustment scheme is devised with the following objectives:

(1) Packet loss: With consideration of future available bandwidth, a TCP connection needs not explore the available bandwidth blindly or rely on packet loss as an indication of congestion. The packet loss incurred should be reduced.

(2) Fairness: The bandwidth of the bottleneck link should be as fairly shared as possible by all

Connection #	10	20	30	40	50	60	70	80	90	100
Hurst para.	0.67	0.71	0.725	0.73	0.745	0.753	0.766	0.77	0.778	0.79

Table 5.1: The Hurst parameter H of the attainable throughput versus the number, N, of connections.

the connections that traverse the link. Note that as indicated in [24] the fairness criterion is 100% met only when all TCP connections are subject to the same RTT.

(3) Stability: The stability of the AIMD algorithm (which has been established in [24] under the assumption of synchronous TCPs) should not be compromised.

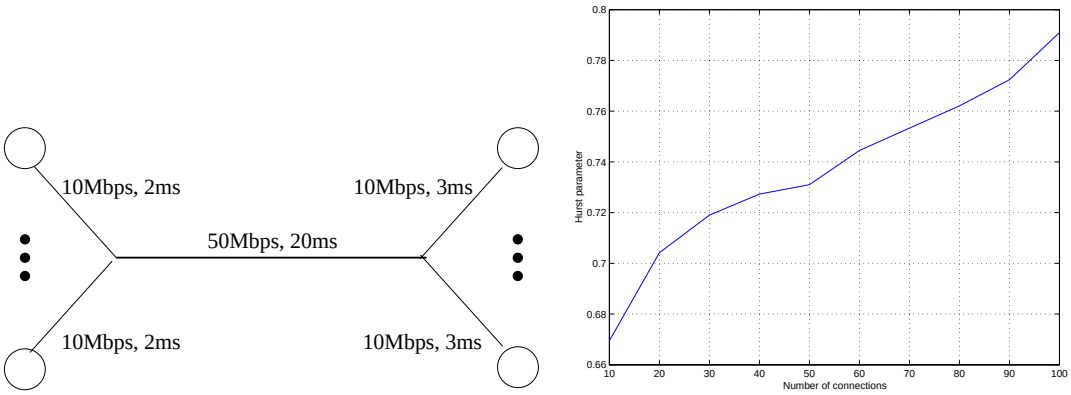
Consider a TCP connection that traverses a path, P_a, from the source to the destination. We characterize the background traffic on the bottleneck link of P_a as a time series, $\{b_t(t), t \in Z+\}$, that exhibits LRD with the Hurst parameter H. We also assume that

(A1) $b_t(t)$, as a superposition of numerous component connections, does not adapt to the TCP connection of interest.

(A2) The capacity (bandwidth), C of the bottleneck link is known. This is not an unreasonable assumption, as several strategies, e.g., the one-packet techniques [74, 115], the packet-pair techniques [104], and a combination thereof [79], have been proposed to measure, without router support, the bandwidth (or cross traffic) of a bottleneck link.

We will discuss in Section 5.2.1 how to relax assumption **(A2)** by approximately inferring the optimal operational point.

With the above setting, it is clear that the bandwidth available to the TCP connection of interest is a time series, $\{X(t) = C - b_t(t), t \in Z+\}$. It can be analytically shown (after simple derivation) that if $b_t(t)$ exhibits LRD, so does $X(t)$ with the same Hurst parameter. To illustrate this, we show in Table. 5.1 the simulation results of a TCP connection sharing a bottleneck link with $N-1$ other connections (which are taken as a whole as the background traffic), where N varies from 10 to 100. The single-bottleneck network topology in which the simulation is conducted is given in Fig. 5.2 (a): The bottleneck link has a capacity of 50 Mbps, a propagation delay of 20ms, and a drop tail queue with a buffer size of 100 packets. TCP packets are generated using



(a) Simulation topology (b) H versus N

Figure 5.2: The Hurst parameter H of the attainable throughput versus the number, N, of connections.

the on-off model in *ns-2*. As shown in Table 5.1, the Hurst parameter of $X(t)$ is in the range of 0.66–0.8, when $N \geq 10$, i.e., $X(t)$ does exhibit LRD. (It has also come to our attention that the authors in [127] conducted a similar study and also showed that $X(t)$ is self-similar.) The fact that $X(t)$ is also self-similar serves the theoretical base of TCP-TP.

TCP-TP follows TCP, and uses non-duplicate ACKs, duplicate ACKs (which serve as NACKs), and timeouts as means of inferring whether or not congestion occurs and the level of congestion. In addition, the sender of a TCP-TP connection keeps track of the amount of data acknowledged and samples the attainable throughput periodically or aperiodically (e.g., whenever a fixed amount of data is acknowledged since the last measurement) using a low-pass filter with an exponentially weighted moving average. That is, if t_0 and t denote the previous and current sampling instants, then

$$X(t) = (1 - \alpha) \cdot X(t_0) + \alpha \cdot \frac{\#\text{ bytes acked since } t_0}{t - t_0}.$$

The sender then keeps track of the time series, predicts the attainable throughput at least one RTT ahead using a *linear minimum mean square error* (*LMMSE*) predictor (has been discussed in Chapter 2), and adjusts its congestion window, with the objective of reaching the optimal operational point without incurring MD phases (to be discussed in Section 5.2.1).

5.2 Detailed Description of TCP-TP

After giving an overview of TCP-TP, we are now in a position to discuss how a TCP sender estimates its future attainable throughput using a *LMMSE*-based predictor and how a TCP connection adjusts, in compliance with prediction results, its congestion window.

5.2.1 Congestion Control with the Use of Prediction Results

Given the time series of the attainable throughput, $\{X(t), t \in Z+\}$, a TCP-TP sender keeps track of the aggregated series, $X^{(m)}(k-n+1), X^{(m)}(k-n+2), \ldots, X^{(m)}(k)$, measured in the past n measurement intervals (Eq. (2.2)). Based on these aggregated series samples, the sender predicts the attainable throughput, $X^{(m)}(k+1)$, in the next measurement interval by applying *LMMSE* predictor discussed in Chapter 2.

Again, we have to determine an appropriate value, τ, for the interval between two calculations of $X^m(k)$. Due to the LRD property, the value of τ is not very critical to performance, and in the simulation study, we set τ to be in the order of one to several RTTs.

We use the phase plot in Fig. 5.1 to describe how the prediction result, $\hat{X}^{(m)}(n+1)$, can be utilized to adjust the window increase/decrease in TCP congestion control. As discussed at the beginning of this chapter,the optimal operational point of N TCP connections sharing the capacity, C, of a bottleneck link is the interaction of line $Tf + TB = C$ (the capacity line) and line $Tf/TB = 1/(N-1)$ (the fairness line).

Suppose the initial operational point is the cross point. Without the knowledge of N, a TCP connection employs the AIMD algorithm, and follows the dashed line to reach the optimal operational point in several round trip times [24]. In the course of reaching the optimal point, the dashed line crosses the capacity line multiple times, implying that multiple packet losses occur (and the MD phase takes effect multiple times). If the TCP connection can infer the value of N (and hence the optimal operational point), it can adjust its congestion window to directly move to the optimal operational point (as shown by the solid line in Fig. 5.1). In the best case, this can be achieved in one RTT without crossing the capacity line (and hence incurring packet losses).

The key issue now is how to infer the number of connections, N. Let $W_i(n)$ and $W_i(n+1)$ denote, respectively, the current congestion window size and the congestion window size in the next

RTT for connection i, and $\eta_i(n)$ denote the ratio of $W_i(n+1)/W_i(n)$. With the estimate of the attainable throughput, $\hat{X}^{(m)}(n+1)$, in the next RTT, and under the fairness criterion, the TCP-TP sender sets

$$N = \left\lceil \frac{C}{\hat{X}^{(m)}(n+1)} \right\rceil. \tag{5.1}$$

TCP-TP then operates as follows. TCP-TP does not change the operations in the slow start phase or the MD method in the congestion avoidance phase. It only changes the AI method in the congestion avoidance phase. That is, when positive acknowledgment is received, a TCP-TP sender sets the congestion window, $W_i(n+1)$, in the next RTT as

$$W_i(n+1) \leftarrow \frac{C \cdot RTT}{N}. \tag{5.2}$$

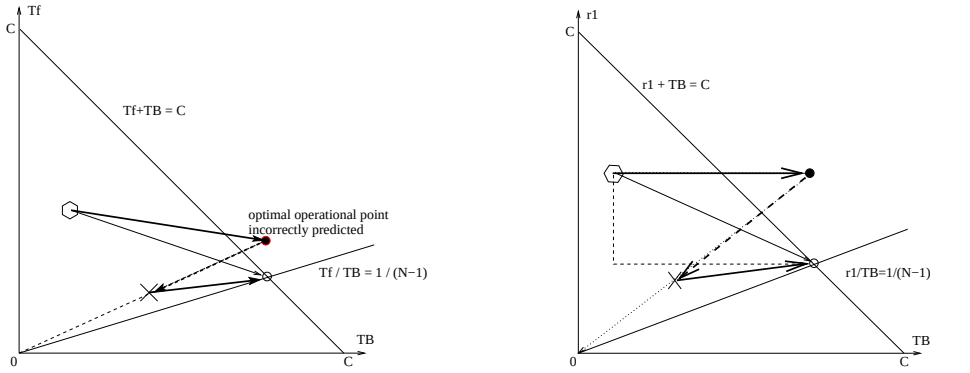
Note that the new adjustment in the AI phase may be (multiplicative) window increase or decrease, depending on the ratio $\eta_i(n)$ is greater than or less than 1. In what follows, we give the correctness claim of TCP-TP under the "idealistic" case in Theorem 6.

Theorem 6: If (i) all TCP connections are subject to the same RTTs, (ii) the initial operational point is below the capacity line, and (iii) the throughput prediction can be accurately made, then the operational point of a TCP-TP connection will not go beyond the capacity line and the TCP-TP connection will reach the optimal operational point in one RTT. *Proof:* Refer to Appendix B.

One point is worthy of mentioning. If the information on the bandwidth, C, of the bottleneck link is not available, we can infer the congestion window, $W_i(n+1)$, in the next RTT by (approximately) combining Eqs. (5.1)-(5.2) as

$$W_i(n+1) \leftarrow RTT \cdot \hat{X}^{(m)}(n+1). \tag{5.3}$$

What if the conditions in Proposition 1 do not hold Now we discuss how TCP-TP operates if one or more of the conditions in Theorem 6 do not hold. If the initial operational point is above the capacity line, the MD phase eventually takes effect and drags the operational point below the capacity line. In the case that prediction errors occur and/or RTTs differ among connections, the



(a) prediction error (b) different RTT among connections

Figure 5.3: The phase plots that illustrate why the operational point occasionally goes beyond the capacity line.

operational point may occasionally go beyond the capacity line and packet loss may occur (Fig. 5.3). For example, as shown in Fig. 5.3 (b), if the TCP-TP connection of interest responds to the system significantly faster (i.e., with a smaller RTT), then the system follows the thin dotted line to reach the optimal operational point. On the other hand, if the "composite" background traffic responds faster, then the system follows the boldfaced line, and may go beyond the capacity line and incur packet loss (at which point the MD phase takes effect). Since we do not modify the MD phase, once packet loss occurs, the congestion window is halved and the operational point is dragged back to below the capacity line (along the dashed line from the filled circle point to the cross point in Fig. 5.3). After that, the revised AI phase takes effect and attempts to drag the operational point toward the optimal point in the next RTT (along the solid line from the cross point to the hollow circle point in Fig. 5.3). As indicated in [24], the MD phase is necessary for TCP to reach equilibrium. By leaving the MD phase unchanged, we ensure that the stability is not impaired by the modification made in the AI phase.

In the case that prediction errors persist, the operational point may never reach the optimal operational point, but instead stagger in the neighboring area of the optimal point. Although the stability is ensured with the use of the MD phase, the long-term fairness may be impaired. We will analytically study the impact of inaccurate prediction on fairness in Section 5.3.

5.2.2 Use of TCP-TP in High Speed Networks

A major concern when TCP-TP is applied to high speed networks in which local access rate is higher than, for example, $1Gbps$ is whether or not the necessary processing at end hosts may become a bottleneck. We claim this will not take place based on the following fact:

1. As compared to traditional TCP algorithms, the major computational overhead of TCP-TP comes from the prediction operation, i.e., the matrix operation in Eq. (2.9). By using the algorithm provided in [30], we can reduce the complexity of the matrix operation to $O(\Omega(n^2))$. Given the fact that n is approximately 20, the computational overhead is not significant. Furthermore, if the algorithm in [1] is used, no matrix operation is needed, and the computational complexity can be reduced to $O(n)$. With the high end PCs (with 4 Ghz processor) currently available, we expect the prediction can be made in nanoseconds in the case of $n = 20$.

2. The prediction operation is not executed very frequently. As stated in Section 5.2, in every τ seconds we update the history information and perform the prediction operation. In both our simulation and empirical studies, we set $\tau = 0.05s$.

5.3 Impact of Prediction Errors on Fairness

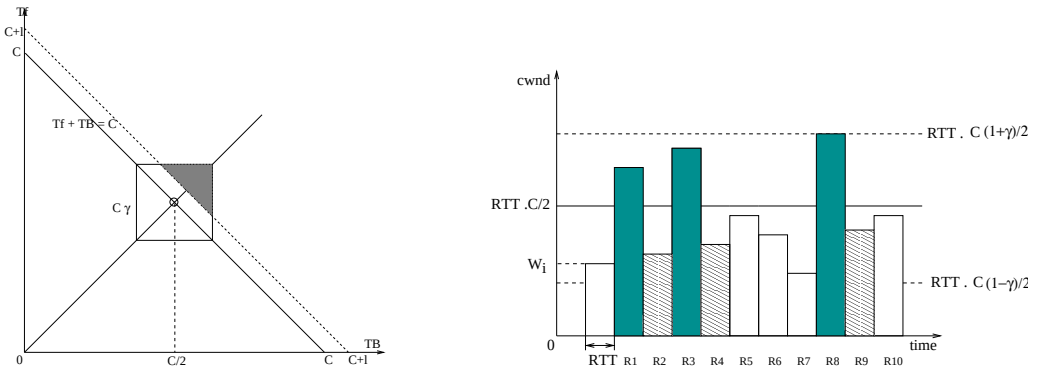
In this section, we analyze how prediction errors affect the performance of TCP-TP in terms of fairness. Specifically, let $\hat{X}_i^{(m)}(n+1)$ and $X_i^{(m)}(n+1)$ denote, respectively, the estimate and actual values of the throughput attained by connection i. Then the prediction error is written as

$$\tau_i(n) = \frac{\hat{X}_i^{(m)}(n+1) - X_i^{(m)}(n+1)}{X_i^{(m)}(n+1)}, \tag{5.4}$$

and the fairness index among N connections (defined in [24]) as

$$F(r) = \frac{(\sum_{i=1}^{N} X_i^{(m)}(n))^2}{N(\sum_{i=1}^{N}(X_i^{(m)}(n))^2)}. \tag{5.5}$$

Note that $F(r) = 1$ if the bandwidth of the bottleneck link is fairly shared among the N connections ($X_i^{(m)}(n) = C/N, \forall i$). We will analyze the impact of $\tau_i(n)$ on $F(r)$ under the cases of $N = 2$ and



(a) phase plot (b) congestion window

Figure 5.4: The phase plot and the time diagram of the congestion window in the case that the prediction error is bounded by γ. W_i is the initial window size.

$N > 2$, respectively.

5.3.1 Case I: $N = 2$

Let $\gamma \stackrel{\triangle}{=} \max(|\tau_1(n)|, |\tau_2(n)|)$. Then, as shown in Fig. 5.4 (a), the predicted optimal operational point falls within a square centered at $(C/2, C/2)$ and with size $C\gamma \cdot C\gamma$. If the buffer size at the bottleneck link is L, then under the assumption that all connections are synchronous, packet loss occurs when the operational point goes beyond line $Tf + TB = C + \frac{L}{RTT}$ (which we call the *augmented capacity line*, Fig. 5.4 (a)). For clarity of notation, let $\ell \stackrel{\triangle}{=} L/RTT$.

For ease of analysis, we assume that the prediction is independently made by each individual connection and that the prediction made in disjoint time intervals is independent of one another. Then the probability that the predicted optimal point falls at any point in the square is uniform over the square. The probability that the predicted optimal point goes beyond the capacity line is then

$$p = \begin{cases} \frac{\text{shaded area in Fig. 5.4 (a)}}{(C\gamma)^2} = \frac{1}{2} \cdot (1 - \frac{\ell}{C\gamma})^2, & \text{if } C\gamma \geq \ell, \\ 0, & \text{if } C\gamma \leq \ell. \end{cases} \tag{5.6}$$

As shown in Fig. 5.4 (b), all the congestion windows under TCP-TP are constrained between the two dashed lines $RTT \cdot C(1+\gamma)/2$ and $RTT \cdot C(1-\gamma)/2$. Furthermore, in the time intervals labeled as $R1$, $R3$, and $R8$, congestion occurs and the congestion window is halved in the following

interval. The long-term attainable throughput T_h can be calculated as

$$\begin{aligned} T_h &= \lim_{M\to\infty} \frac{\sum_{n=1}^{M} W_i(n)}{M \times RTT} \\ &= \lim_{M\to\infty} \frac{M_c \cdot E(W_c) + M_d \cdot E(W_d) + M_u \cdot E(W_u)}{M \times RTT}, \end{aligned} \tag{5.7}$$

where M_c, M_d, and M_u are, respectively, the number of intervals in which congestion occurs, the number of intervals immediately following an interval in which congestion occurs, and the number of the other intervals, and $E(W_c)$, $E(W_d)$, and $E(W_u)$ are, respectively, the average window sizes in the M_c, M_d, and M_u intervals.

Derivation of M_c, M_d, and M_u Based on the uniform distribution assumption, with probability p (Eq. (5.6)) the predicted optimal point goes beyond the augmented capacity line and congestion occurs. Following the interval in which congestion occurs, the MD phase takes effect, the congestion window is halved, and the operational point falls below the augmented capacity line. Then, in the next interval a new prediction is made and the cycle repeats. Consequently,

$$M_c = M_d = \frac{p}{1+p} \cdot M; \quad M_u = \frac{1-p}{1+p} \cdot M. \tag{5.8}$$

Derivation of $E(W_c)$, $E(W_d)$, and $E(W_u)$: To facilitate derivation of $E(W_c)$, $E(W_d)$, and $E(W_u)$, we first give the following two lemmas:

Lemma 3: Given two random variables, X and Y, with the joint probability density function

$$f(x,y) = \begin{cases} \left(\frac{(b-a)^2}{2}\right)^{-1}, & \text{if } (x,y) \in \text{shaded triangle in Fig. 5.5(a)}, \\ 0, & \text{otherwise.} \end{cases}$$

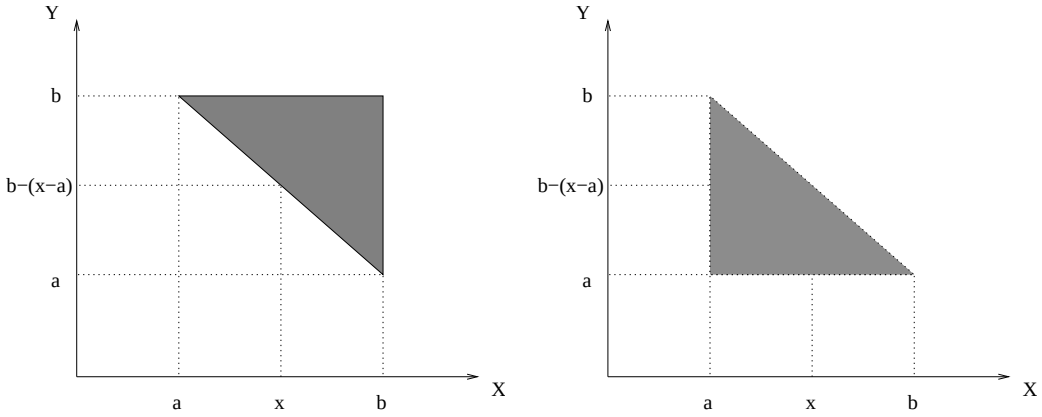
Then $E(X) = E(Y) = \frac{a+2b}{3}$.

Lemma 4: Given two random variables, X and Y, with the joint probability density function

$$f(x,y) = \begin{cases} \left(\frac{(b-a)^2}{2}\right)^{-1}, & \text{if } (x,y) \in \text{shaded triangle in Fig. 5.5(b)}, \\ 0, & \text{otherwise.} \end{cases}$$

Then $E(X) = E(Y) = \frac{2a+b}{3}$.

In the case that $C\gamma \le \ell$, the operational point always falls within the augmented capacity line, no packet loss occurs ($p = 0$), and hence $T = C/2$ and $F = 1$. Fig. 5.6 depicts the enlarged version



(a) p.d.f. for Lemma 1 (b) p.d.f. for Lemma 2

Figure 5.5: Probability density functions of two random variables, X and Y.

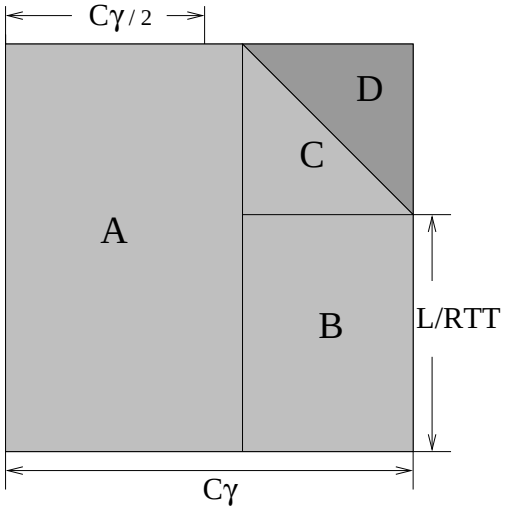
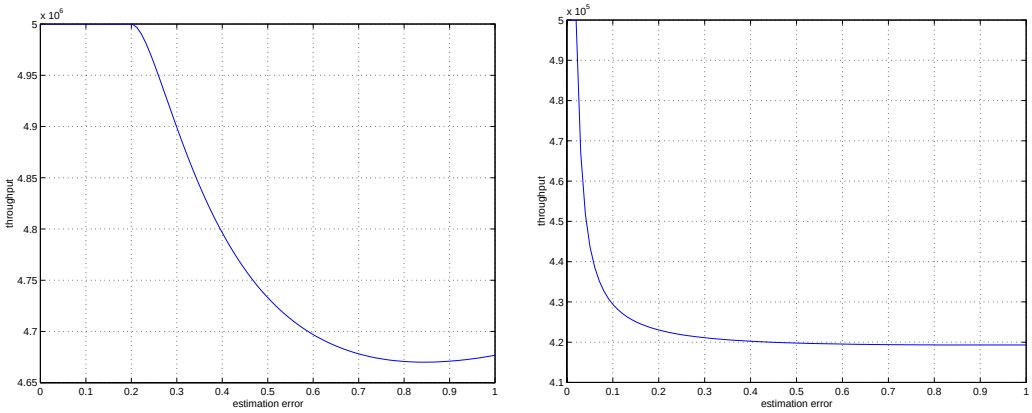


Figure 5.6: An enlarged version of the square that characterizes the prediction error in the case that the prediction error is bounded by γ, the buffer size is L, and $N = 2$.



(a) $N = 2$ (b) $N = 20$

Figure 5.7: Long-term attainable throughput, T_h, versus maximum percentage of prediction errors, γ.

of the square that characterizes the prediction error in the case of $C\gamma \geq \ell$. When the operational point falls in the shaded area D, packets loss occurs. Under the uniform distribution assumption and using Lemma 3, $E(W_c)$ can be written as

$$\begin{aligned} E(W_c) &= \frac{RTT}{3} \cdot ((\frac{C - C\gamma}{2} + \ell) + 2(\frac{C}{2} + \frac{C\gamma}{2})) \\ &= \frac{RTT \cdot ((3+\gamma)C + 2\ell)}{6}. \end{aligned} \tag{5.9}$$

Since following the interval in which congestion occurs, the congestion window is halved, the average window size, $E(W_d)$, is simply

$$E(W_d) = \frac{E(W_c)}{2} = \frac{RTT \cdot ((3+\gamma)C + 2\ell)}{12}. \tag{5.10}$$

Derivation of $E(W_u)$ is a little bit more complicated. As shown in Fig. 5.6, the operational point could be in areas A, B or C. With a little bit of geometric reasoning, it is straightforward to show (i) with probability $P_A = \frac{\text{area in A}}{\text{area in square}} = \frac{2C\gamma\ell}{C^2\gamma^2 + 2C\gamma\ell - \ell^2}$, the operational point falls in area A, and the window size is uniformly distributed in $[\frac{C}{2} - \frac{C\gamma}{2}, \frac{C}{2} - \frac{C\gamma}{2} + \ell]$; (ii) with probability $P_B = \frac{2C\gamma\ell - 2\ell^2}{C^2\gamma^2 + 2C\gamma\ell - \ell^2}$, the operational point falls in area B, and the window size is uniformly distributed in $[\frac{C}{2} - \frac{C\gamma}{2} + \ell, \frac{C}{2} + \frac{C\gamma}{2}]$; and (iii) with probability $1 - P_A - P_B$ the operational point falls in area C, and the congestion window size can be derived using Lemma 4 as $\frac{RTT \cdot ((3-\gamma)C + 4\ell)}{6}$. $E(W_u)$ can

then be written as

$$\begin{aligned} E(W_u) &= P_A \cdot RTT \cdot \frac{(\frac{C}{2} - \frac{C\gamma}{2} + \ell) + (\frac{C}{2} - \frac{C\gamma}{2})}{2} \\ &\quad + P_B \cdot RTT \cdot \frac{(\frac{C}{2} - \frac{C\gamma}{2} + \ell) + (\frac{C}{2} + \frac{C\gamma}{2})}{2} \\ &\quad + (1 - P_A - P_B) \cdot \frac{RTT \cdot (3C + 4\ell - C\gamma)}{6}. \end{aligned} \tag{5.11}$$

By plugging Eqs. (5.8) and (5.9)–(5.11) in Eq. (5.7), we have

$$\begin{aligned} T_h &= \frac{1}{12(1+p)} \cdot (p((3+5\gamma)C - 2\ell) + 2(1-p)(C\gamma(P_B - 2P_A) \\ &\quad -\ell(P_A + P_B)) + (6 - 2\gamma)C + 8\ell), \end{aligned} \tag{5.12}$$

where $P_A = \frac{2C\gamma\ell}{C^2\gamma^2 + 2C\gamma\ell - \ell^2}$ and $P_B = \frac{2C\gamma\ell - 2\ell^2}{C^2\gamma^2 + 2C\gamma\ell - \ell^2}$.

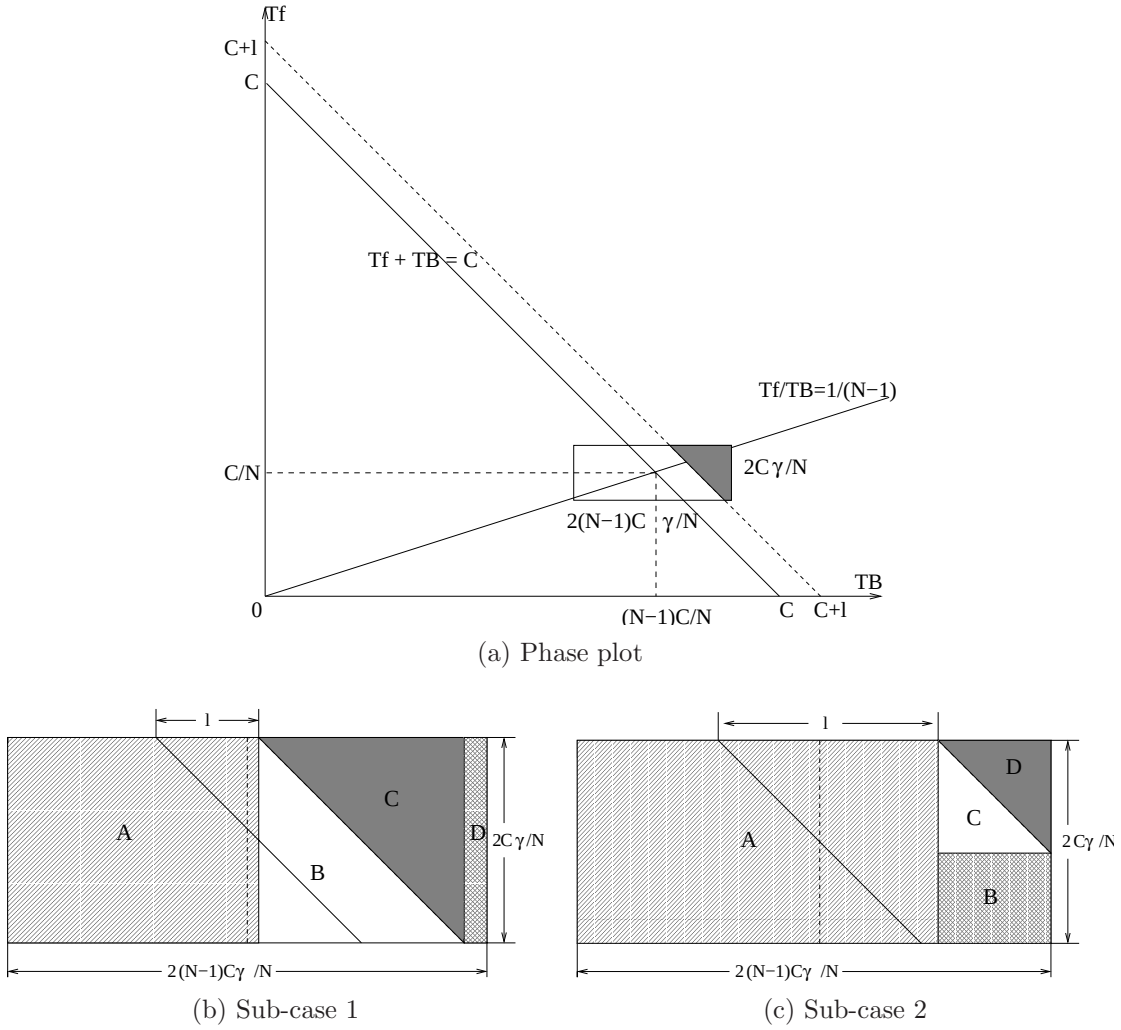
An example plot of T versus γ is given in Fig. 5.7 (a), where $C = 10$ Mbps, $L = 40$ packets with the average packet size of 1000 bytes, and RTT = 20 ms. As γ increases from 0 to 1, T decrease from $C/2 = 5$ Mbps to 4.65 Mbps. That is, F decreases from 1 to 0.995, or a 0.5% degradation in fairness in the case of the maximum percentage of prediction error $\gamma = 1$.

5.3.2 Case II: $N > 2$

Fig. 5.8 gives the phase plot in the case that the prediction error is bounded by γ and $N > 2$. The predicted optimal operational point now falls in a rectangle with edges equal to $\frac{2C\gamma}{N}$ and $\frac{2(N-1)C\gamma}{N}$. When the operational point falls in the shaded area, packet loss occurs. As shown in the enlarged version of the rectangle that characterizes the prediction error in Fig. 5.8 (b)–(c), we have to consider the following three sub-cases.

Sub-Case 1 In this case, we have $0 \le \ell \le \frac{N-2}{2N} \cdot C \cdot \gamma$ and the rectangle area can be divided into four areas, labeled as A, B, C, and D in Fig. 5.8 (b). When the operational point falls in area C or D, packet loss occurs, the probability of which can be derived, with a little bit geometrical reasoning, as $p = \frac{(N-1)C\gamma - N\ell}{2(N-1)C\gamma}$. Following the same line of reasoning in Section 5.3.1, we also have $M_c = M_d = \frac{p}{1+p} \cdot M$, and $M_u = \frac{1-p}{1+p} \cdot M$.

To derive $E(W_c)$ (and $E(W_d)$), note that when packet loss occurs, (i) with probability $P_C = \frac{C\gamma}{(N-1)C\gamma - N\ell}$, the operational point falls in area C, and the average window size is $RTT \cdot \frac{2(\frac{C}{N} + \frac{C\gamma}{N}) + (\frac{C}{N} - \frac{C\gamma}{N})}{3} =$



(a) Phase plot

(b) Sub-case 1

(c) Sub-case 2

Figure 5.8: The phase plot in the case that the prediction error is bounded by γ and $N > 2$.

$RTT \cdot \left(\frac{C}{N} + \frac{C\gamma}{3N}\right)$ (Lemma 3); and (ii) with probability $1 - P_C$, the operational point falls in area D, and the average window size is $RTT \cdot \frac{C}{N}$. Thus, $E(W_c)$ can be written as

$$E(W_c) = P_C \cdot RTT \cdot (\frac{C}{N} + \frac{C\gamma}{3N}) + (1 - P_C) \cdot RTT \cdot \frac{C}{N}, \tag{5.13}$$

and

$$E(W_d) = \frac{E(W_c)}{2} = RTT \cdot \frac{C}{2N} \cdot (1 + \frac{1}{3}\frac{C\gamma^2}{(N-1)C\gamma - 2N\ell}). \tag{5.14}$$

To derive $E(W_u)$, note that when packet loss does not occur, (i) with probability $P_A = \frac{C\gamma}{(N-1)C\gamma + N\ell}$, the operational point falls in area A, and the average window size is $RTT \cdot \frac{C}{N}$; and (ii) with probability $1 - P_A$, the average window size is $RTT \cdot \frac{\frac{C}{N} + \frac{C\gamma}{N} + 2(\frac{C}{N} - \frac{C\gamma}{N})}{3} = RTT \cdot (\frac{C}{N} - \frac{C\gamma}{3N})$ (Lemma 4). Thus $E(W_u)$ can be written as

$$E(W_u) = P_A \cdot RTT \cdot \frac{C}{N} + (1 - P_A) \cdot RTT \cdot (\frac{C}{N} - \frac{C\gamma}{3N}). \tag{5.15}$$

Finally, the long-term attainable throughput T can be expressed as:

$$\begin{aligned} T_h = & \frac{1}{2(1+p)} \cdot \frac{C}{N} \cdot (2 + p + \frac{C\gamma^2}{3}(\frac{3p}{(N-1)C\gamma - N\ell} \\ & - \frac{2(1-p)}{(N-1)C\gamma + N\ell})). \end{aligned} \tag{5.16}$$

Sub-Case 2 In this case, we have $\frac{(N-2)C\gamma}{N} \leq \ell < C\gamma$ and the rectangle area can also be divided into four areas (Fig. 5.8 (c)). When the operational point falls in area D, packet loss occurs, the probability of which can be derived, following the same line of reasoning in sub-case 1, to be $p = \frac{N^2(C\gamma - \ell)^2}{8(N-1)C^2\gamma^2}$. Following the same line of reasoning in Section 5.3.1, we also have $M_c = M_d = \frac{p}{1+p} \cdot M$, and $M_u = \frac{1-p}{1+p} \cdot M$.

Using Lemma 3, $E(W_c)$ can be written as:

$$\begin{aligned} E(W_c) &= RTT \cdot \frac{2(\frac{C}{N} + \frac{C\gamma}{N}) + (\frac{C}{N} + \ell - \frac{(N-1)C\gamma}{N})}{3} \\ &= RTT \cdot \frac{3C + N\ell + (3-N)C\gamma}{3N}, \end{aligned} \tag{5.17}$$

and hence $E(W_d) = \frac{E(W_c)}{2}$ can be expressed as

$$E(W_d) = RTT \cdot \frac{3C + N\ell + (3-N)C\gamma}{6N}. \tag{5.18}$$

To derive $E(W_u)$, note that (i) with probability $P_C = \frac{N^2(C\gamma-\ell)^2}{8C^2\gamma^2(N-1)-N^2(C\gamma-\ell)^2}$ the operational point falls in area C, and the average window size is $RTT \times \frac{(\frac{C}{N}+\frac{C\gamma}{N})+2(\frac{C}{N}+\ell-\frac{(N-1)C\gamma}{N})}{3}$ (Lemma 4); (ii) with probability $P_B = \frac{2N^2(\ell-\frac{(N-2)C\gamma}{N})(C\gamma-\ell)}{8C^2\gamma^2(N-1)-N^2(C\gamma-\ell)^2}$, the operational point falls in area B, and the average window size is $RTT(\frac{C}{N}+\frac{\ell}{2}-\frac{C\gamma}{2})$; and (iii) with probability $1-P_C-P_B$, the operational point falls in area A, and the average window size is $RTT \cdot \frac{C}{N}$. Thus $E(W_u)$ can be written as

$$\begin{aligned} E(W_u) &= (1-P_B-P_C)\cdot RTT\cdot\frac{C}{N} + P_B\cdot RTT\cdot(\frac{C}{N}+\frac{\ell}{2}-\frac{C\gamma}{2}) \\ &\quad + P_C\cdot RTT\cdot\frac{\frac{C}{N}+\frac{C\gamma}{N}+2(\frac{C}{N}+\ell-\frac{(N-1)C\gamma}{N})}{3}. \end{aligned} \tag{5.19}$$

Finally the long-term attainable throughput T_h can be expressed as

$$\begin{aligned} T_h &= \frac{C}{(1+p)2N}\times(2+p+\frac{Np\ell}{C}+(3-N)p\gamma+\frac{(1-p)\ell N}{3C}\times \\ &\quad (3P_B+4P_C)+\frac{(1-p)\gamma}{3}((4-2N)P_C-3NP_B)). \end{aligned} \tag{5.20}$$

Sub-Case 3 In this case, we have $\ell \le C\gamma$ and the operational point always falls within the augmented capacity line, no packet loss occurs ($p=0$), and hence $T=\frac{C}{N}$ and $F=1$.

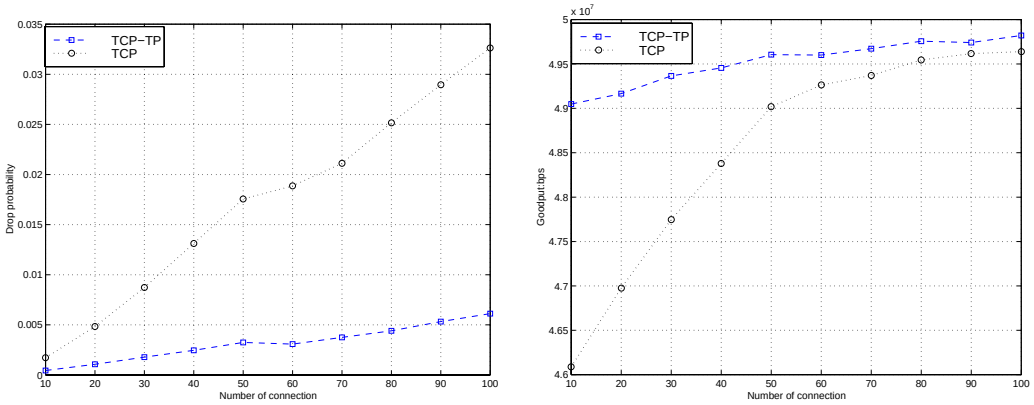
Fig. 5.7 (b) gives an example plot of T versus γ, with $C=10$ Mbps, $N=20$, $L=40$ packets with the average packet size of 1000 bytes, and RTT = 20 ms. As γ increases from 0 to 1, T decreases from $C/N=0.5$ Mbps to 0.42 Mbps and F decreases from 1.0 to 0.975 (approximately 2.5% degradation). This conclusion also holds true for other combinations of C, N and L. This demonstrates that the impact of prediction errors on fairness is minimal in the cases of both $N=2$ and $N>2$.

5.4 Performance Evaluation

We have implemented TCP-TP both in ns-2 and in the FreeBSD 4.1 kernel, and conducted simulation and empirical studies to validate the proposed design and compared the performance of TCP-TP against TCP-new Reno.

5.4.1 Simulation Results

In the simulation study, we examine the behavior of TCP-Reno and TCP-TP under a variety of network topologies (e.g., the single bottleneck topology, the multiple bottleneck link topology (e.g.,



(a) Packet loss ratio (b) Attained throughput

Figure 5.9: Performance of TCP-new Reno and TCP-TP w.r.t. packet loss rate and attained throughput in the case of equal RTTs.

Fig. 5.16), and arbitrary network topology) and scenarios. Unless otherwise specified, the *LMMSE* predictor is used to estimate the attainable throughput in the next interval. Each data point is the result averaged over 10 simulation runs. In what follows we report on a small set of simulations which we believe is most representative.

Results in the Single-Bottleneck Topology

The single-bottleneck network topology used is the same as that used in Fig. 5.2. We establish N TCP connections that generate packets (of size 1000 bytes) under the on-off traffic model, where N varies from 10 to 100. We then measure the packet loss ratio, the throughput attained by all TCP receivers, and the congestion window of each TCP connection.

Whether or not LRD diminish if all connections use TCP-TP One interesting question is whether or not TCP-TP is "self-defeating," i.e., whether or not the LRD characteristic diminishes if all the connections use TCP-TP as their transport layer protocol. In the first experiment, we investigate this question by using, for all the N connections, the on-off model (with the shape parameter $\alpha = 1.5$) as the traffic source and TCP-TP as the transport layer protocol. All connections are subject to the same RTT = 50 ms. Fig. 5.10 depicts the calculated Hurst parameter under the above scenario.

As shown in Fig. 5.10, the traffic still exhibits LRD. This is attributed to the fact that the

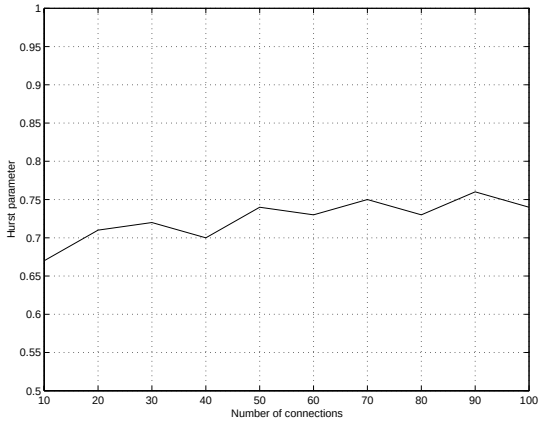


Figure 5.10: The Hurst parameter as a function of the number TCP-TP connections.

congestion control algorithm used in the transport layer protocol is not the unique cause of LRD. As reported in [33] and [130], the presence of heavy tails in lengths of individual flows can be shown to induce self-similarity. The authors of [33] also reported that the distribution of file sizes, the effects of caching and human factors like response time and preference are possible causes for self-similarity in WWW traffic, as it can be shown that file sizes, human thinking time and flow (session) duration all have heavy-tailed property.

Performance under the case of equal RTTs In the second experiment, we assume the same scenario as in the first experiment except that either TCP-new Reno or TCP-TP is used as the transport layer protocol. Fig. 5.9 gives the performance of TCP-new Reno and TCP-TP. TCP-TP outperforms TCP-new Reno both in terms of packet loss ratio and throughput attained by all receivers. In particular, the performance improvement in packet loss ratio can be as high as 75%. The performance improvement in the total attained throughput is not as significant. This is because the attained throughput is constrained by the bandwidth of the bottleneck link (evidenced by the decreasing performance gap between TCP-TP and TCP as N increases). To further characterize the cause of performance gain, we depict in Fig. 5.11 the instantaneous window size of one of the connections. TCP-TP incurs much less window reductions as compared to TCP-new Reno. This shows that with the judicious use of prediction results, TCP-TP can greatly reduce the likelihood that the operational point goes beyond the capacity line. We also calculate the fairness, F, under both TCP-TP and TCP-new Reno. The results are both in the range of 0.98–0.99, and are very

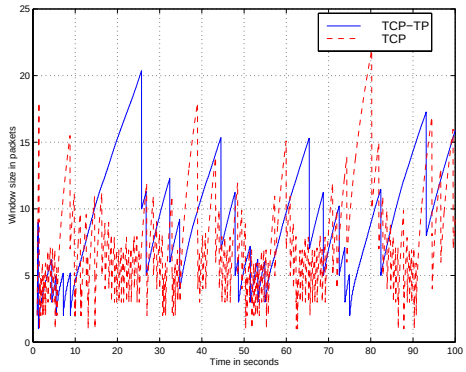


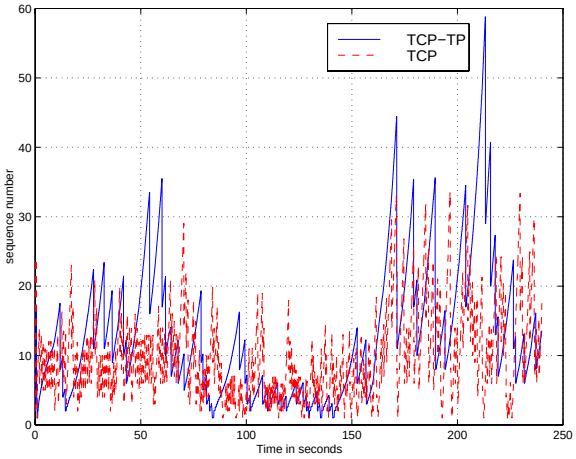
Figure 5.11: The instantaneous window size of a connection in Experiment 1.

close to each other.

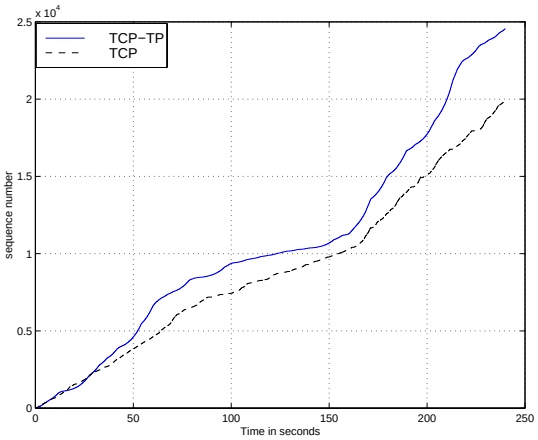
Performance under dynamic changes of TCP connections In the third experiment, we use the same topology as in the first experiment, but dynamically establish/terminate connections. Initially 30 connections commerce at time 0s. At time 100s, additional 30 connections are established. At time 160s, 40 connections are terminated, while the others continue until the end of the simulation (240 s). We measure the instantaneous window size and the sequence number increment of a connection that commences at time 0s and runs until 240s.

Fig. 5.12 gives the instantaneous window size, the increment in sequence number, and the derivative of the sequence number increment under both TCP-TP and TCP-new Reno. As shown in Fig. 5.12 (a), TCP-TP responds more quickly to the change of N at time 100s and 160s, meaning that it reaches the new optimal operational point faster. This is corroborated by the observation in Fig. 5.12 (c): around time 100s, TCP-TP has a smaller derivative of sequence number, implying that the sequence number increases slower in response to the establishment of new connections. A similar conclusion can be made at time 160s at which TCP-TP has a larger derivative of sequence number, implying that the sequence number increases faster. Also, as shown in Fig. 5.12 (b), the sequence number under TCP-TP is always larger than that under TCP, i.e., TCP-TP attains a better goodput.

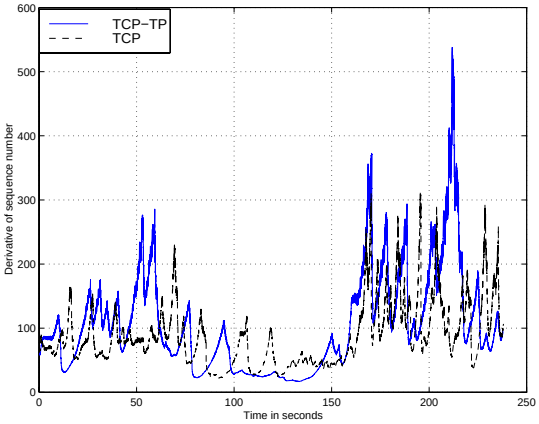
Performance in the case of different RTTs In the fourth experiment, we use the same network topology as in the first experiment, but vary the RTT experienced by different connections. The RTT of a TCP connection is drawn from an uniform distribution $[20, 100]$ ms. Fig. 5.13 gives



(a) Instantaneous window size

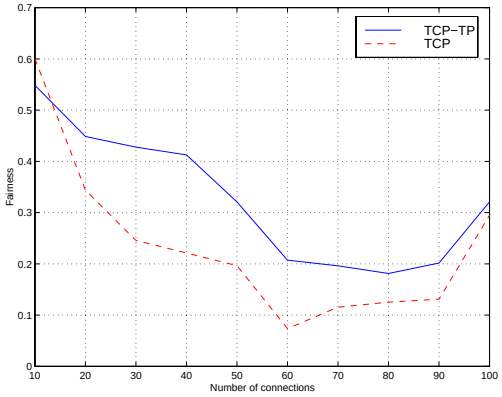


(b) Sequence number

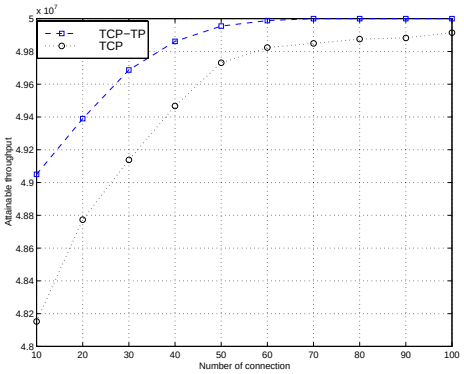


(c) Derivative of SN

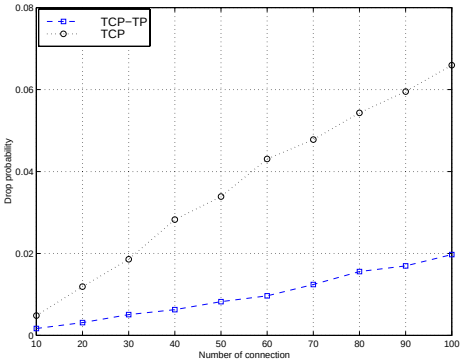
Figure 5.12: Performance of TCP-new Reno and TCP-TP in the case of dynamic connection establishment and termination.



(a) Fairness



(b) Attainable throughput

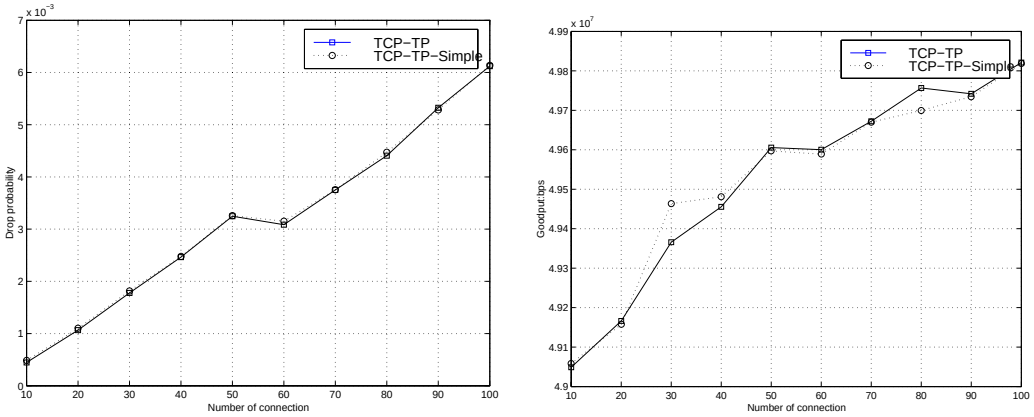


(c) Packet loss ratio

Figure 5.13: Performance of TCP-new Reno and TCP-TP in the case of different RTTs.

the performance of TCP-new Reno and TCP-TP. The performance shows a similar trend to that under the synchronous TCP case (in the second experiment), except with respect to fairness. As the fairness analysis is conducted under the assumption of equal RTTs, both TCP and TCP-TP suffer in the case of different RTTs. Nevertheless, TCP-TP still achieves (30%) better fairness than TCP, suggesting TCP-TP is less susceptible to the effect of RTTs. This can be attributed to the fact that TCP-TP attempts to drag the operational point *directly* to the optimal point that achieves fairness.

Performance comparison between *LMMSE* and *simple* In this experiment, we compare *LMMSE* against *simple*. We use the same simulation setting as in Experiment 1, but use both *LMMSE* and *simple* predictors to infer the attainable throughput in the next interval. Fig. 5.14 depicts the the packet loss ratio and the attainable throughput under *LMMSE* (denoted as *TCP-*



(a) Packet loss ratio (b) Attainable throughput

Figure 5.14: Performance of TCP-TP and TCP-TP-Simple with respect to packet loss ratio and attainable throughput.

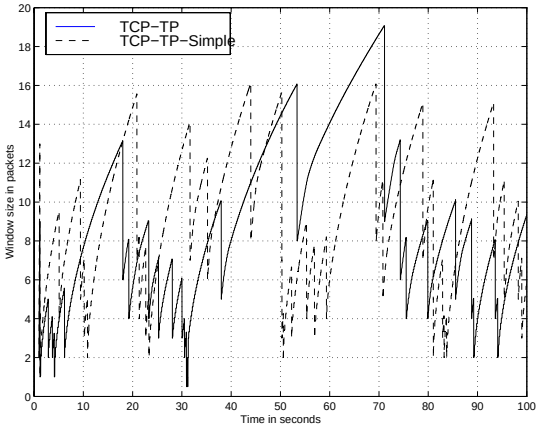


Figure 5.15: Instantaneous window size under TCP-TP and TCP-TP-Simple.

TP) and *simple* (denoted as *TCP-TP-Simple*). Fig. 5.15 gives the corresponding instantaneous window size.

As shown in Fig. 5.14, *TCP-TP-Simple* achieves almost the same performance as *TCP-TP*. With respect to attainable throughput, when the number of connections is small (≤ 50), *TCP-TP-Simple* achieves slightly higher throughput than *TCP-TP*, while for a large number of connections (> 50), the reverse is true. Their performance with respect to packet loss ratio is almost indistinguishable. As the *simple* predictor incurs much less computational overhead, *TCP-TP-Simple* is a better choice when the CPU cycles are scare.

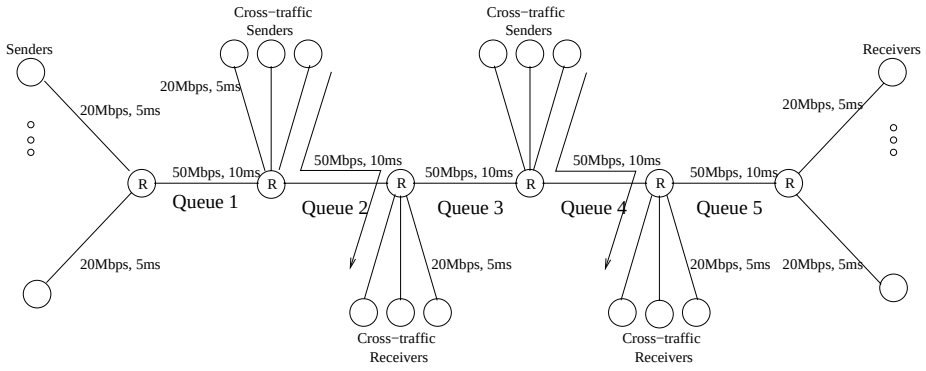


Figure 5.16: The multiple bottleneck simulation topology.

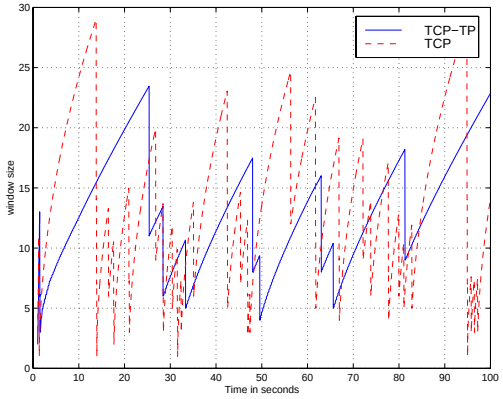


Figure 5.17: Performance of TCP-new Reno and TCP-TP w.r.t. congestion window size in the case of multiple bottleneck links.

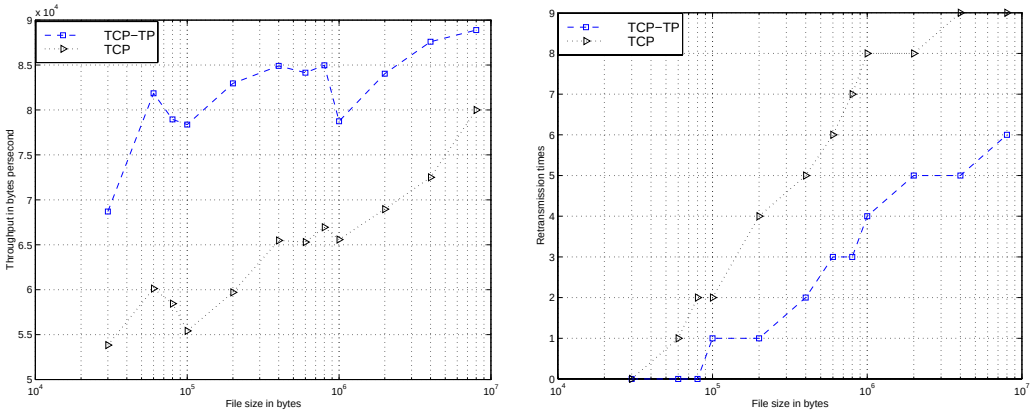
Results in the Multiple-Bottleneck Topology

The multiple-bottleneck topology used is shown in Fig. 5.16, in which five queues exist on the end-to-end path. Cross traffic is generated between the second and third routers (queue 2), and between the forth and the fifth routers (queue 4). We establish N end-to-end TCP connections, and $0.5N$ TCP connections in each cross traffic bundle, where N varies from 10 to 100. All the TCP connections generate packets using the on-off traffic model. The performance is very similar to that of the single bottleneck case. Hence we only depict in Fig. 5.17 the curve of the instantaneous window size under TCP-New Reno and TCP-TP.

5.4.2 Empirical Study

We have implemented TCP-TP in the FreeBSD 4.1. kernel. Succinctly, the state machine, the data structure and the various TCP modules are described in detail in [132]. The major changes we made are that we include a *LMMSE* predictor in the *TCP-Input* module. In particular, we added several data structures into the TCP protocol control block to keep track of the recent history of the amount of successfully transmitted (acknowledged) data. This data is updated by the *TCP-input* function whenever an non-duplicate acknowledgment is received. In addition, the window increase/decrease operations in the AI phase of the congestion avoidance phase is modified. We did not explicitly measure the bandwidth, C, of the bottleneck link using tools such as *Pathcar* [74], but instead use Eq. (5.3) to infer the congestion window size in the next time interval, thus bypassing the need to obtain the value of C. The implementation requires an addition/change of approximately 100 lines of C code into the kernel. Although this version of implementation is on the FreeBSD kernel, it is straightforward to port it to other UNIX-based operating systems.

We carried out experiments over the Internet, with receivers located at UCSB (alpha.ece.ucsb.edu), UCI (rodan.ics.uci.edu), UMD (dsp7.eng.umd.edu), and UW-Madison (hertz.ece.wisc.edu) and the sender located at OSU (eepc118.eng.ohio-state.edu). The experiments were carried out in 3 different time intervals (morning, afternoon, and night) on a daily basis for a period of 2 weeks. In each set of experiments, we establish a HTTP connection between the sender and a receiver, and either TCP or TCP-TP is used as the underlying transport protocol. The size of the file to be transferred varies from 30 Kbytes (which is the average size of a web page) to 8Mbytes (which represents extremely large file transfer). We measure the average throughput for each connection and the total number of retransmission timeouts occurred (the later is an indication of multiple, consecutive packet losses). Fig. 5.18 gives the empirical results for experiments performed between OSU and Wisconsin in the afternoon intervals. (The other results exhibit similar trends and hence are not shown.) Again TCP-TP outperforms TCP with respect to the average throughput and the number of retransmission timeouts.



(a) Average throughput (b) Retransmission timeouts

Figure 5.18: Empirical results of experiments between OSU and UW.

5.5 Summary

We have demonstrated in this chapter that the self-similar characteristics of network traffic can be exploited to increase throughput gains and reduce packet losses in TCP congestion control. In particular, we show that with the use of a simple *LMMSE* predictor, one can accurately (with estimation error $\leq 15\%$) estimate the future traffic at least one RTT ahead. A TCP connection can then use the prediction result to infer the optimal operational point at which a TCP connection should operate. Although the analysis is pinpointed in the context of rather complicated AIMD steady-state dynamics, the resulting scheme (called *TCP-TP*) is light weight, requires modification (tens of lines of code change) only at the TCP sender side, and achieves performance gains (in some simulation cases, up to 75%) in terms of packet loss ratio, attainable throughput, and responsiveness to dynamic network traffic changes.

Chapter 6

Traffic Measurement

In this chapter we present three theoretically grounded methods: prediction, reconstruction and interpolation, for measuring cross traffic on the bottleneck link of an end-to-end path. The objective is to infer cross traffic as accurately as possible, while not injecting a significant amount of probe packets into the network. In the prediction-based method, we take advantage of the *LRD* characteristic of the cross traffic to predict the future traffic based on the recent information obtained by probe packets. In the reconstruction method, we rebuild the entire cross traffic process with the information obtained by probe packets. In the interpolation method, we periodically send closely-spaced probe packet pairs to sample cross traffic of the bottleneck link, and infer cross traffic between two sampling points using interpolation. The simulation study indicates that (i) the prediction-based and reconstruction methods can give good mean measurement of cross traffic, while the interpolation method usually captures the instantaneous value of cross traffic better; and (ii) all three methods are adaptive to the dynamic change of cross traffic and are quite robust in the presence of multiple bottleneck links on an end-to-end path. We also did empirical study by implementing the proposed methods at both user and kernel level.

The rest of the chapter is organized as follows. In Section 6.1, we state the assumptions made and describe the pattern of probe packets used in this chapter. Then we delve into the detailed description of the three proposed methods in Sections 6.2–6.4. Following that, we present simulation and empirical results in Section 6.5 and Section 6.6. Last, we conclude the chapter in Section 6.7.

6.1 Preliminary

6.1.1 Assumptions and Probe Packets Pattern

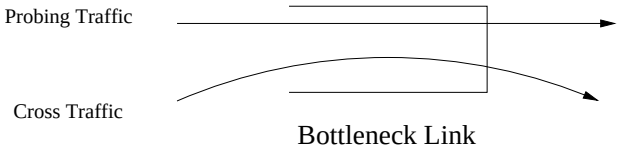


Figure 6.1: Single bottle neck link model.

We consider an end-to-end path with a bottleneck link. Closely-spaced probe packets are sent along the path, and the time interval between arrivals of two consecutive packets at the destination is measured and used to infer the cross traffic (Fig. 6.1). The assumptions made in this chapter are: (**A1**) there exists only one bottleneck link on the end-to-end path; (**A2**) packets will not be queued before or after the bottleneck link; and (**A3**) the capacity of the bottleneck link is known. Under these assumptions, the volume of cross traffic entering the queue between two probe packets can be exactly computed. This is because under (**A2**) the time interval between these two probe packets does not change after they leave the bottleneck link until they arrive at the destination. Without (**A2**) (i.e., if the two probe packets experience queuing delay at some queues other than the queue at the bottleneck link), the time interval between arrivals of the two packets at the destination may be stretched or squeezed, leading to underestimate or overestimate of the cross traffic. Although (**A3**) is reasonable[1], the first two assumptions may not hold true in real networks. In Section 6.5, we will study through *ns-2* simulation the robustness of our methods if these two assumptions do not hold.

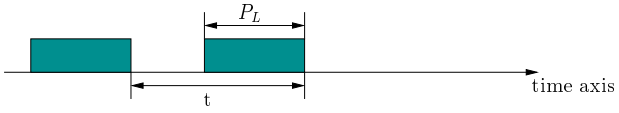


Figure 6.2: Two back-to-back probe packets.

The temporal pattern of probe packets is shown in Fig. 6.2. At the sender, two closely-spaced probe packets of length P_L (called a *probe packet pair*) are sent. Suppose the interval between their

[1]One can use traceroute to determine the path on which the packets traverse and find out the type each link (T-1, T-3, or OC-n) on the path; alternatively, one may use the end-to-end measurement technique in [78, 79] to infer the bottleneck bandwidth.

sending times is t. If there exists no cross traffic at the bottleneck link, and $t < \frac{P_L}{C}$, where C is the bottle-neck link capacity, then the dispersion of the arrival times of the two packets $t^{'}$ at the destination satisfies:

$$t^{'} = \frac{P_L}{C} \tag{6.1}$$

or

$$C = \frac{P_L}{t^{'}} \tag{6.2}$$

Eqs. (6.1)–(6.2) are accurate under the condition: $t < \frac{P_L}{C}$. Since with capacity C, at most P_L traffic can be served in $\frac{P_L}{C}$ seconds, before the first packet leaves the queue, the second packet has been queued, the time interval between the arrivals of the two probe packets is exactly $\frac{P_L}{C}$.

Let the amount of cross traffic that arrive during the time interval $[t_0, t_0+t]$ be denoted as $X_t(t_0)$, where t_0 is the time instant the first packet of the probe packet pair traverses the bottleneck link. Then the dispersion of the arrival times of the probe packet pair at the destination is:

$$\tau_d = \frac{P_L + X_t(t_0)}{C}, \tag{6.3}$$

or

$$X_t(t_0) = C\tau_d - P_L. \tag{6.4}$$

Since the time interval τ_d can be measured exactly at the receiver, with the knowledge of C and P_L, we can infer the cross traffic that arrive in interval $[t_0, t_0 + t]$.

A naive method to obtain the volume of cross traffic at any time is to send constantly closely-spaced probe packets and measure the dispersion of arrival times of consecutive probe packets. However, since C is usually large and P_L is small as compared to C, the value of t has to be very small. For example, if $C = 2$Mbps, $P_L = 1000$ bytes, then $t < 4ms$. That is, a large amount of probe packets have to be sent and the bandwidth of the bottleneck link will be consumed mainly by probe packets. In this chapter, we will devise three methods to accurately infer the amount of cross traffic without sending a large amount of probe packets.

n	5	10	15	20	30	40	50
$err(\%)$	14.2	11.5	9.3	7.8	7.7	7.7	7.6

Table 6.1: Relative prediction error for different values of n.

6.2 The Prediction-Based Method

In the prediction-based method, the sender sends $n+1$ closely-spaced probe packets (with the temporal distance between the sending times of two consecutive packets being t). The destination measures the inter-arrival times of the $n+1$ probe packets, and obtains n samples of τ_d. The time series $X_t(t)$ with length n can then be constructed using Eq. (6.4), and furthermore the aggregated time series $X_a(k), k = 1, 2, ...n$ can be obtained by dividing $X_t(t)$ by t.

Based on these aggregate series samples, we predict the future cross traffic using the Linear Minimum Mean Square Error (*LMMSE*) estimator (discussed in Chapter 2). Specifically, given the series $X_a(k), k = 1, \ldots, n$, the aggregate cross traffic series in the next time interval t, $X_a(n+1)$, can be expressed as a weighted linear combination of the past n samples, where the weights are determined to minimize the mean square error. That is, the estimate (written as $\hat{X}_a(n+1)$) of $X_a(n+1)$ is expressed as

$$\hat{X}_a(n+1) = \begin{bmatrix} a_1 & a_2 & ... & a_n \end{bmatrix} \begin{bmatrix} X_a(1) \\ X_a(2) \\ ... \\ X_a(n) \end{bmatrix}, \tag{6.5}$$

where $a_1, a_2, ..., a_n$ are the *LMMSE* coefficients

Both He *et al.* [57] and Sang *et al.* [114] have shown that the above estimator can make very good prediction of $\hat{X}_a(n+1)$. After $\hat{X}_a(n+1)$ is predicted, $\hat{X}_a(n+2)$ can be predicted by using $X_a(k), k = 1, 2, ..., n$ and $\hat{X}_a(n+1)$, $\hat{X}_a(n+3)$ can be predicted by using $X_a(k), k = 1, 2, ..., n$, $\hat{X}_a(n+1)$ and $\hat{X}_a(n+2)$, and so on. The process continues until N predictions are made.

Determination of tunable parameters: There are two tunable parameters that we need to determine: one is the number, n, of closely-spaced packets that are sent at the beginning, and the other is the number, N, of predictions that can be made before prediction accuracy is impaired as

N	n	$2n$	$3n$	$4n$	$5n$	$6n$	$8n$
$err(\%)$	7.8	8.5	9.1	10.2	11.8	15.4	19.8

Table 6.2: Relative prediction error for different values of N.

a result of accumulated prediction errors in each step.

To determine the value of n, we have conducted *ns-2* simulation in which traffic traces are generated on a dumbbell topology and on arbitrary network topologies and the above *LMMSE*-based approach is used to predict the future traffic. Table 6.1 gives the relative prediction error for different values of n under the dumbbell topology given in Fig. 6.8 (a), where the number of TCP connections varies from 10 to 100, and the relative error is defined as $\frac{|\hat{X}_a(t)-X_a(t)|}{X_a(t)}$. (The result under arbitrary topologies are similar and hence omitted.) As shown in Table 6.1, the larger the value of n, the smaller the relative error (i.e., the more accurate the prediction result). However, the performance improvement levels off as n exceeds 20. This is due to the fact that $R(\tau)$ decreases quite dramatically, and hence adding more history information can not further improve the prediction accuracy. In the simulation study we set $n = 20$.

To determine the value of N, we also conduct the same set of experiments (with n set to 20) to study the effect of varying the value of N on the prediction accuracy. The experiment results are shown in Table 6.2. When N grows beyond $5n$ the relative prediction error is more than 15%. Hence, in the simulation study, we set $N = 5n$, i.e., the sender sends $n + 1$ closely-spaced probe packets at the beginning and infers the amount of cross-traffic in the next $5n \times t$ period. After that, the sender sends another $n + 1$ probe packets, and the entire process repeats.

6.3 The Reconstruction Method

Similar to the traffic prediction method, the sender sends $n + 1$ closely-spaced probe packets, and obtains the time series $X_a(k), k = 1, 2, ..., n$. The time series is then used to reconstruct the *entire* process under the assumption that the amount of cross traffic is statistically stationary. Conceptually, we obtain the autocorrelation function $R^m(k)$ of $X_a(k)$ using Eq. (2.10), and estimate the power spectral density, $p(s)$, of $X_a(k)$, i.e., the *Fourier* transform of $R^m(k)$. Since the power spectral density is the square of the *Fourier* transform of the original time series, $X_a(k)$ can be

obtained in principle using the inverse *Fourier* transform of the square root of $p(s)$.

To practically implement this method, we consider two issues: (i) how to get the estimate of the autocorrelation function $R(k)$; and (ii) how to reconstruct the process $X_a(k)$. We leverage the method by Davis *et al.* [11]. Succinctly, given a Gaussian, zero-mean time series of length n with autocorrelation function $r(i), i = 0, 1, ..., n-1$,[2], we perform the following operations:

1. Define the finite *Fourier* transform g_k of the sequence $r(0), r(1), ..., r(n-2), r(n-1), r(n-2), ..., r(1)$ as follows. Let

$$\omega_k \triangleq \frac{2\pi(k-1)}{(2n-2)}, \tag{6.6}$$

for $k = 1, 2, ..., 2n-2$, and

$$R(i) \triangleq \begin{cases} r(i), & i = 0, 1, ..., n-1, \\ r(2n-2-i), & i = n, n+1, ..., 2n-3. \end{cases} \tag{6.7}$$

Note that for any real WSS (wide sense stationary) random process, its autocorrelation function is even, and we use the one sided autocorrelation function $r(i)$ to generate $R(i)$. Then, the finite *Fourier* transform g_k is defined as

$$g_k = \sum_{j=0}^{2n-3} R(j) \cdot e^{ij\omega_k}, k = 1, 2, ..., 2n-2. \tag{6.8}$$

2. Generate two independent series of zero mean normal random variables, $U_1, U_2, ..., U_n$ and $V_2, ..., V_{n-1}$, such that $var(U_1) = var(U_n) = 2$ and for $k \neq 1, n$, $var(U_k) = var(V_k) = 1$. Let $V_1 = V_n = 0$ and define complex random variables Z_k as:

$$Z_k = \begin{cases} U_k + iV_k, & k = 1, 2, ..., n \\ U_{2n-k} - iV_{2n-k}, & k = n+1, ..., 2n-2. \end{cases} \tag{6.9}$$

3. For $t = 1, 2, ..., n$, define

$$X_t = \frac{1}{2\sqrt{n-1}} \sum_{k=1}^{2n-2} \sqrt{g_k} e^{i(t-1)\omega_k} Z_k. \tag{6.10}$$

[2]Under the stationary assumption, one can transform an arbitrary process into a process with zero mean by subtracting the mean from the process, and adding the mean back at the end of this algorithm.

X_t corresponds to the time series, $X_a(k)$, we would like to reconstruct. The rationale behind constructing a complex random variables Z_k is as follows. For any real WSS random process, its autocorrelation function is a deterministic function of time lag, τ, and the corresponding Fourier transform function g_k is also deterministic (due to the fact that $R(\tau)$ is even). The process is not reversible, i.e., after taking the square root of g_k, we obtain a deterministic signal, whose inverse Fourier transform is also deterministic (i.e., the recovered signal is no longer random). To reconstruct a random variable we have to rely on another random variable with the uniform distribution. Here, Z_k plays this role. Multiplying $\sqrt{g_k}$ with Z_k with is equivalent to convoluting $r(\tau)$ with a constant (due to the fact that Z_k is white). In this way, we introduce randomness to the reconstructed process while keeping the autocorrelation structure unchanged.

Note that in step 3 a n-point inverse *Fourier* transform is performed to reconstruct the original process. Since the original process exhibits self similarity, its autocorrelation structure is asymptotically the same at different time scales and we can envision the reconstructed process as the amount of aggregated cross traffic at any time scale. For example, if we envision the reconstructed process as an aggregated process at the time scale of $2t$, we can get the estimate in the period of $2nt$. In theory as long as the process is self similar, the sender needs only to send $n + 1$ probe packets and can obtain estimates in the entire time domain. However, in reality since the process is not strictly self-similar, and the autocorrelation structure estimated based on the $n + 1$ probe packets may introduce estimate error, the amount of cross-traffic can not be estimated at arbitrarily large time scales. In other words, $n + 1$ probe packets have to be sent every Nt period, and the values of both n and N have to be determined.

Determination of tunable parameters: We have conducted the same set of *ns-2* simulation runs as in Section 6.2 to study the effect of varying the value of n on the reconstruction error. The result is similar to that obtained in the traffic prediction method, i.e., as n goes beyond 20, the reduction in the reconstruction error levels off. This is not a coincidence, as both methods depend heavily on the autocorrelation structure of the time series. Henceforth, in the simulation study we set $n = 20$.

To determine the value of N, we have to consider two issues: accuracy and computation complexity. Again we conduct the same set of experiments as in Section 6.2. As shown in Table 6.3, the

N	n	$2n$	$3n$	$4n$	$5n$	$6n$	$8n$
$err(\%)$	8.7	9.2	9.7	10.6	11.3	13.4	15.1

Table 6.3: Relative reconstruction error for different values of N.

larger the value of N, the more pronounced the reconstruction error but the lower the computation complexity ($n+1$ probe packets per Nt time units). In the simulation study we set $N = 4n$, partially due to the facts that the reconstruction error is $\leq 10\%$ and that the factor $4 = 2^2$ facilitates application of fast Fourier transform algorithm.

6.4 The Interpolation-Based Method

Both the traffic prediction and reconstruction methods require periodic sending of $n+1$ closely-spaced probe packets. The computational complexity in the traffic reconstruction method is also non-negligible, although fast Fourier transform algorithm can be used. In addition, both methods can only capture the mean value of the time series of interest. In this section, we propose an interpolation-based method that requires a smaller number of probe packets and yet gives better estimates. In what follows, we first give an overview of this method, and then delve into the discussion of implementation details.

6.4.1 Overview

According to the *Nyquist* criterion, a signal can be reconstructed as long as the rate at which the signal is sampled is at least twice as large as the bandwidth of the signal. As the cross traffic exhibits long-range dependency, we have $\sum_{k=0}^{\infty} R(k) = \infty$, and hence the power spectral density $p(s) \to \infty$ as $s \to 0$. In other words, the power spectrum of the cross-traffic has the $\frac{1}{f}$ property, i.e., the cross traffic has a much narrower bandwidth (and hence requires a much smaller sampling rate) as compared to traditional Gaussian white noise.

The power spectral density satisfies

$$p(s) \sim \text{const} \cdot s^{\beta-1}, \tag{6.11}$$

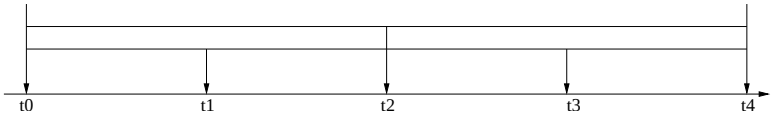


Figure 6.3: Use of the cross traffic information obtained at t_0 and t_4 to interpolate the cross traffic at the three middle points t_1, t_2, and t_3.

where $0 < \beta < 1$ is some constant and is related to the Hurst parameter through $H = 1 - \frac{\beta}{2}$. As the typical value of the Hurst parameter for Internet traffic is $H = 0.8$, we have $\beta = 0.4$. Since $p(s)$ goes to infinity at $s = 0$, we can not calculate the $3dB$ bandwidth as usual. Instead, we calculate the bandwidth when $p(s)$ drops to $\frac{1}{2} \cdot const$ (denoted as B_w), where $const$ is the constant defined in Eq. (6.11). Note that B is much larger than $3dB$ bandwidth. By Eq. (6.11), we have $B_w = 3.2$, i.e., if we sample the original signal at the rate $r \geq 6.4$ or the sampling interval $T \leq 156ms$, we can reconstruct the original signal without incurring error. (We will demonstrate later that T can be even larger, due to the self-similarity of the signal.)

In the end-to-end measurement problem considered, the signal (the amount of cross traffic on the bottleneck link) is sampled by having the sender send a pair of closely-spaced probe packets every T seconds. Each pair of such packets gives one sample of aggregated cross traffic $X_a(t)$ at time t. Then we employ the interpolation-based method to rebuild the time series. Consider, for example, Fig. 6.3: the sender sends a pair of probe packets at time instants t_0 and t_4 ($T = t_4 - t_0$). Then the information obtained at t_0 and t_4 is used to interpolate the cross traffic at the middle point between t_0 and t_4. Similarly, after the information at t_2 is obtained, it is used (along with information obtained at t_0 and t_4) to interpolate the cross traffic at time instants t_1 and t_3. This process repeats recursively until a desirable time granularity is reached.

There are two issues that must be considered in order to implement the interpolation-based method: (i) how to determine the value of the sampling cycle T; and (ii) how to interpolate the amount of cross traffic with the cross traffic information available at the two endpoints. We will elaborate on the second issue in the next subsection, and defer the discussion of the first issue to Section 6.5.

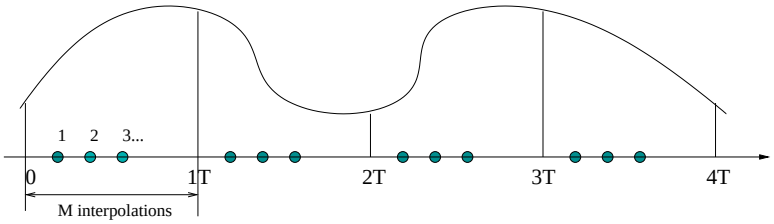


Figure 6.4: The input signal to the FIR filter with m interpolated points.

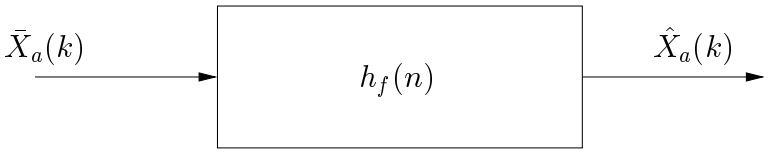


Figure 6.5: The FIR filter.

6.4.2 Implementation of the Interpolation-Based Method

Fig. 6.3 demonstrates that one can interpolate, with the use of the cross traffic information obtained at the two endpoints, the amount of cross traffic at the middle point of an interval T. As a matter of fact, by designing an appropriate finite impulse response (*FIR*) filter one can interpolate the amount of cross traffic at I_M points that are evenly distributed in an interval T. Specifically, let $X_a(k)$ be the cross traffic in the discrete time domain as defined before. With the sampled values of $X_a(k)$ at time T, $2T$, ..., we would like to interpolate the amount of cross traffic at the I_M evenly-spaced points between any two sampled values.

The interpolation procedure with the use of *FIR* filters is as follows. As shown in Fig. 6.4, we first set the values at the I_M points in each interval T to be 0, and pass the sequence, $\overline{X}_a(k)$, with I_M zero values inserted, to the *FIR* filter. The output of the *FIR* filter, $\hat{X}_a(k)$, is the interpolated signal (Fig. 6.5). The impulse response of the filter $h_f(n)$ is designed to have the following properties:

1. $h_f(n)$ is a time series of length $2I_M + 1$.
2. $h_f(-i) = h_f(i) = a_i, 1 \leq i \leq I_M$, and $h_f(0) = 1$.

Since $\hat{X}_a(k) = \overline{X}_a(k) \star h_f(k)$, where $\star$ is the convolution operation, it is straightforward to see that $\hat{X}_a(iT) = \overline{X}_a(iT), i = 0, 1, 2, ...$ This is desirable, because the output should be the same as the input at sampling points. The symmetric design of $h_f(n)$, on the other hand, guarantees that the FIR filter has the linear phase and hence a constant time delay can be guaranteed.

Next we need to determine the values of a_i's in order to achieve the minimum mean square error. By the definition of convolution, the i-th interpolated value can be represented as:

$$\hat{X}_a(k+i) = a_{I_M-i+1}\overline{X}_a(k) + a_i\overline{X}_a(I_M+k+1), i = 1, 2, ..., I_M. \tag{6.12}$$

To fulfill the minimum mean square error criterion, we have

$$E((X_a(k+i) - \hat{X}_a(k+i)) \cdot \hat{X}_a(k+i)) = 0. \tag{6.13}$$

After some algebraic operations, Eq. (6.13) gives

$$\begin{aligned} &a^2_{I_M-i+1}R(0) + a_i^2R(0) + 2a_ia_{I_M-i+1}R(I_M+1) \\ &\quad -a_{I_M-i+1}R(i) - a_iR(I_M+1-i) = 0, i = 1, 2, ..., I_M, \end{aligned} \tag{6.14}$$

where $R(k)$ is the autocorrelation function of $X_a(k)$. To determine the values of a_i's, we have to estimate the autocorrelation structure $R(k)$ of the cross traffic $X_a(k)$. To this end, in the interpolation-based method we enable the sender to send $n+1$ closely-spaced probe packets to get n samples initially, so that we can estimate $R(k)$ using Eq. (2.10). Since only $R(k)$, $0 \le k \le I_M$, are needed, and I_M is usually small (≤ 5), we can get accurate estimates of $R(k)$, even if n is not large. In contrast to the prediction-based and reconstruction methods, the interpolation-based method does not require that the sender sends periodically $n+1$ probe packets. After the sender sends $n+1$ probe packets initially, it needs only to send one pair of probe packets every T seconds.

Note that Eq. (6.14) contains I_M equations with I_M unknowns. However, because the coefficients are symmetric, the I_M equations are not independent, and the I_M coefficients cannot be uniquely determined. In order to determine the values of the I_M coefficients, we have to introduce some other relation among the coefficients. (We will discuss this further below.) Next we consider two special cases: $I_M = 1$ and $I_M = 2$.

$I_M = 1$: The impulse response of the filter $h_f(n)$ has the following property: (i) $h_f(n)$ is a time series of length 3; and (ii) $h_f(-1) = h_f(1) = \alpha$, and $h_f(0) = 1$. What is left to determine is

$\alpha = h_f(1)$. By Eq. (6.12), we have

$$\begin{aligned} \hat{X}_a(k) &= \alpha\overline{X}_a(k-1) + \alpha\overline{X}_a(k+1) \\ &= \alpha X_a(k-1) + \alpha X_a(k+1). \end{aligned} \tag{6.15}$$

Also, to achieve minimum mean square error, $X_a(k) - \hat{X}_a(k)$ should be perpendicular to $\hat{X}_a(k)$, i.e.,

$$E((X_a(k) - \hat{X}_a(k)) \cdot \hat{X}_a(k)) = 0, \tag{6.16}$$

or,

$$\alpha^2 R(0) + \alpha^2 R(2) - \alpha R(1) = 0. \tag{6.17}$$

Hence,

$$\alpha = \frac{R(1)}{R(0) + R(2)}. \tag{6.18}$$

$I_M = 2$**:** The impulse response of the filter $h_f(n)$ is a time series of length 5 and with two unknown coefficients. Let $a_f = h_f(2) = h_f(-2)$ and $b_f = h_f(1) = h_f(-1)$ denote the two coefficients of the *FIR* filter yet to be determined. Using the similar technique as above we obtain

$$a_f^2 R(0) + b_f^2 R(0) + 2a_f b_f R(3) - a_f R(2) - b_f R(1) = 0. \tag{6.19}$$

Due to the fact that coefficients are symmetric, the other equation is the same as Eq. (6.19). Hence, we have to add another condition. To enforce a linear decrease in the coefficients, we set $b_f = \frac{1+a_f}{2}$. Other relations between a_f and b_f are also possible. For example, we may enforce the coefficients to decrease exponentially from 1 to a_f. We have conducted simulations to investigate the impact of the additional condition on the performance, and found that in all the simulation runs the performance is rather insensitive to the condition as long as b_f is larger than a_f. Thus, we choose $b_f = \frac{1+a_f}{2}$ so as to obtain closed form results:

$$a_f = \frac{-c_2 + \sqrt{c_2^2 - 4c_1c_3}}{2c_1}, \tag{6.20}$$

Where $c_1 = 5R(0) + 4R(3)$, $c_2 = 2R(0) + 4R(3) - 4R(2) - 2R(1)$, and $c_3 = R(0) - 2R(1)$.

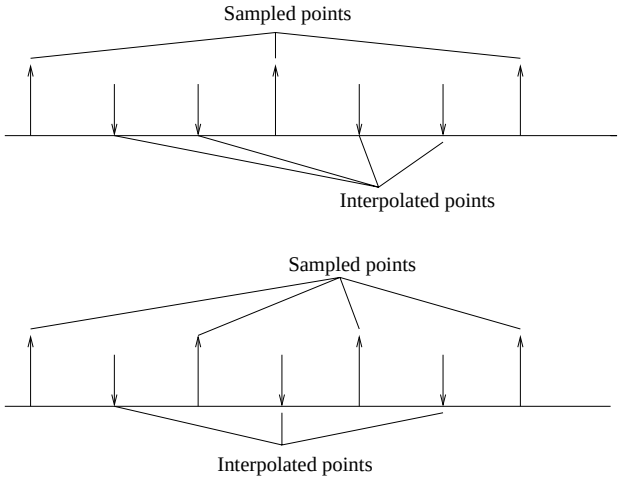


Figure 6.6: An example that shows the difference between interpolating the signal at 1 or 2 points between the two given samples.

Discussion: Note that the choice of I_M represents a trade-off between the ability to interpolate the signal at more points between the two given samples, the accuracy of the interpolation results, and the complexity of the resulting *FIR* filter. As shown in Fig. 6.6, if the interpolation is performed at one point ($I_M = 1$), 4 pairs of probe packets are needed; on the other hand, if the amount of cross traffic is interpolated at two points between the two given samples ($I_M = 2$), only 3 pairs of probe packets need to be sent. However, the saving in the number of probe packets does not come without a cost. The *FIR* filter for $I_M = 2$ is more complex.

We have conducted the same set of *ns-2* simulation runs as in Section 6.2 to study the effect of varying the value of I_M on the interpolation error. We have also implemented the FIR filter in *Matlab* for both $I_M = 1$ and $I_M = 2$. Fig. 6.7 gives the relative mean interpolation error. Contrary to our intuition, the interpolation error is smaller under the case of $I_M = 2$. This can be explained as follows. In the case of $I_M = 2$, the first interpolated value(and so is the second interpolated value), I_n, is $\frac{2T}{3}$ time units away from the right sample N_A, and $\frac{T}{3}$ time units away from the left sample N_B, and I_n is calculated as $I_n = a_f \cdot N_A + b_f \cdot N_B$, where $a_f < b_f$, a_f and b_f are the parameters of the FIR filter in Eq. (6.19). As N_B is temporally closer to I_n, the estimate of I_n is more accurate by giving more weight to N_B. In contrast, in the case of $I_M = 1$, equal weights are assigned to both the two samples, N_A and N_B, that are $\frac{T}{2}$ time units away. One should, however,

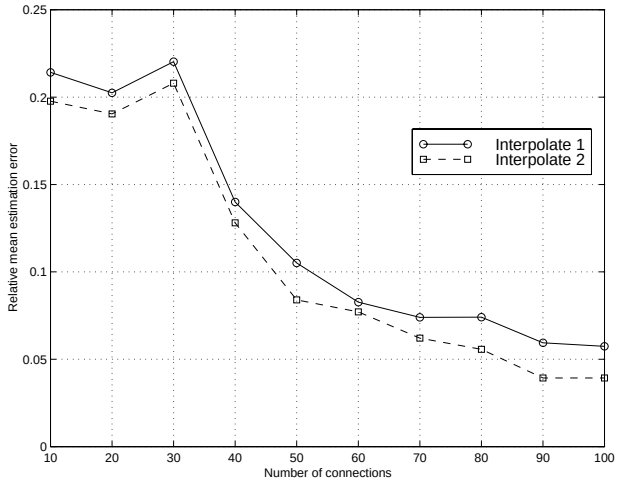


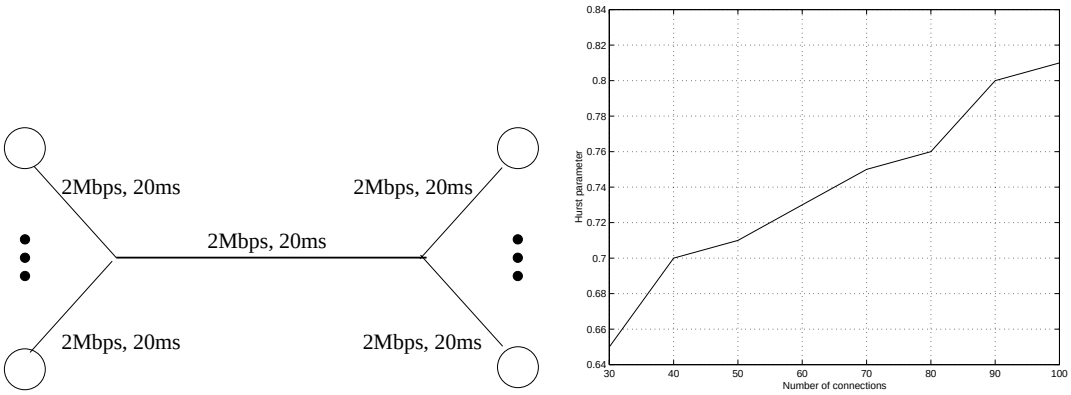
Figure 6.7: The mean interpolation error in the cases of $I_M = 1$ and $I_M = 2$.

not generalize on this result, i.e., it may not be true that the larger the value of I_M, the more accurate the interpolation results will be. This is because as I_M increases, the design of the FIR filter becomes more difficult, and the interpolation results will be very sensitive to the coefficients. In the simulation study, we implement FIR filters with $I_M = 1$ and $I_M = 2$.

6.5 Simulation Results

We have implemented the prediction-based, reconstruction, and interpolation-based methods in *ns-2* and conducted a simulation study to validate the proposed design and compare the performance. The performance metrics of interest are (i) relative mean error $err = \frac{\hat{X}_a(t) - X_a(t)}{X_a(t)}$, (ii) standard deviation of the error *std*, (iii) ability of adapting to the changes of traffic load on the bottleneck link, and (iv) robustness in the case that some of the assumptions (**A2**) are relaxed.

We examine the behavior of these methods under a variety of network topologies and traffic sources. In particular, we have considered the network topologies with a single bottleneck link, with multiple bottleneck links, as well as arbitrary topologies. The maximum buffer size of each router is set to 100 packets (each of size 1000 bytes). The interval between two consecutive, back-to-back probe packets is set to be $0.005s$. We have used an assortment of traffic sources (e.g., TCP sources that generate packets according to the on-off model or real traffic traces down-loaded from the Internet, and constant bit rate UDP sources) as the sources of cross traffic. Each data point is the



(a) Dumbbell topology (b) Hurst parameter

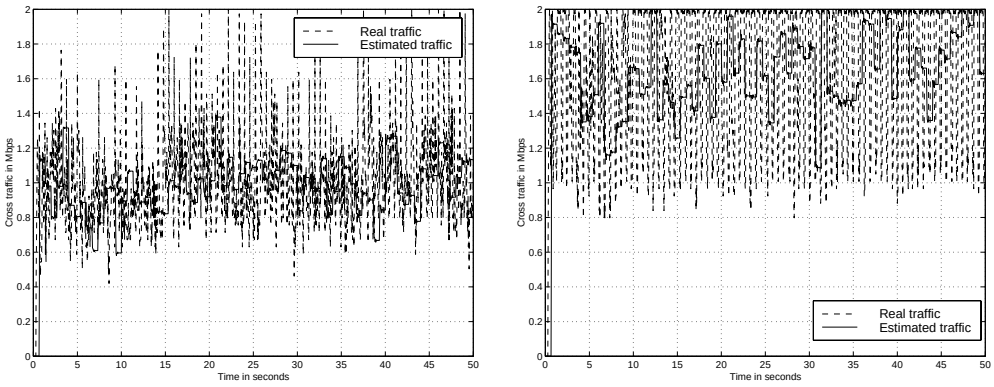
Figure 6.8: Simulation topology and the Hurst parameter of cross traffic in Experiment 1.

result averaged over 10 simulation runs, and each simulation run lasts for 50 seconds. We report on a small set of the simulations which we believe is the most representative. In spite of numerous system parameters involved, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves for a representative set of parameter values (reported below) is valid over a wide range of parameter values.

6.5.1 Experiment 1: Performance under Different Link Utilizations

In the first set of experiments, we evaluate the performance of the three methods under different link utilizations. We also study the effect of varying the value of T (the interval between two samples) on the performance of the interpolation-based method.

The network topology used (along with all the relevant network parameters) is shown in Fig. 6.8 (a). The cross traffic on the bottleneck link is made up of 30-100 UDP/TCP connections, with the left-hand-side (right-hand-side) hosts being the sources (destinations), and with *Pareto* on-off models (with the shape parameter $\alpha = 1.5$) being the traffic generation models. The end-to-end measurement is performed by another source-destination pair that send probe packets in compliance with the method under consideration. As shown in Fig. 6.8 (b), the *Hurst* parameter H of the cross traffic ranges from 0.65 to 0.81, as the number of connections increases from 30 to 100, indicating that the cross traffic does exhibit the LRD characteristics. If there were no cross traffic, the interval t should satisfy: $t \leq \frac{P_L}{C} = \frac{8000}{2000000} = 0.004s$. In the presence of cross traffic, t can be set to be a



(a) 30 sources (b) 50 sources of cross traffic

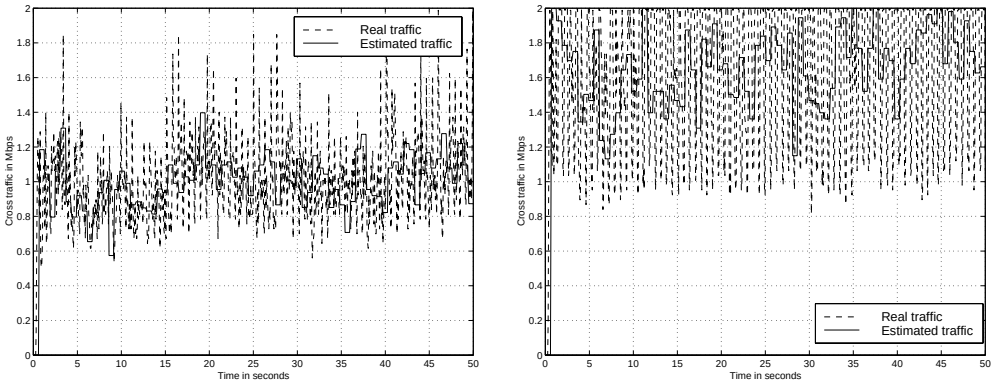
Figure 6.9: Cross traffic estimated using the prediction-based method versus actual traffic in the existence of 30 and 50 sources of cross traffic.

little larger. We keep track of the cross traffic and its estimate.

In the prediction-based method, 21 back-to-back probe packets are sent at the beginning of each interval of length 0.6 second (120 times t). Since it takes $20 \times 0.005 = 0.1s$ to send 21 back-to-back probe packets, the prediction-based method estimates the amount of cross traffic for the remaining 0.5 second. As mentioned in Section 6.2, this amount of time is equal $5nt = 100 \times 0.005$. In the reconstruction method, again 21 back-to-back probe packets are sent at the beginning of each interval of length 0.5 second. An 80-point inverse *Fourier* transform is performed to reconstruct the original cross traffic process in an time interval of of length $4nT = 0.4s$. In the interpolation-based method, 21 back-to-back probe packets are sent initially. Following that, in every T seconds 2 back-to-back probe packets are sent. The amount of cross traffic in the T interval is estimated by interpolation. Note that the choice of T will have an impact on the performance of the interpolation-based method, as there exists a trade-off: the smaller the value of T, the smaller the relative mean error; however, as more probe packets have to be sent, the larger the standard deviation of the error. We have experimented with different values of T, and will show below the results in the cases of $T = 0.05$ s and $T = 0.5$ s.

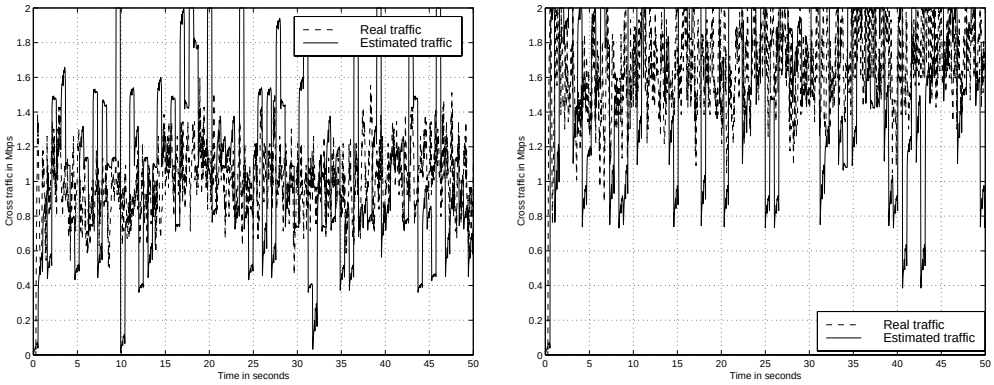
Figs. 6.9–6.34 give the simulation results under the prediction-based, reconstruction, and interpolation-based methods, respectively, in the existence of 30 and 50 sources of cross traffic. Tables 6.4–6.6 give *err* and *std* under the three methods. Several observations are in order:

- As shown in Figs. 6.9–6.10, both the prediction-based and reconstruction methods capture



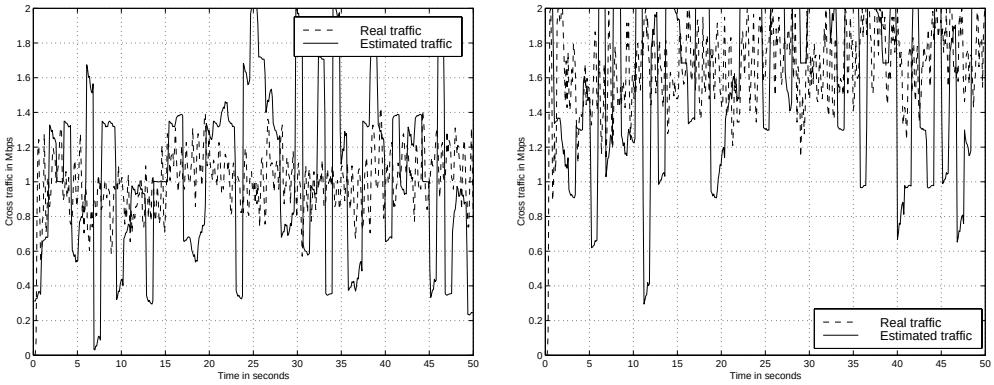
(a) 30 sources (b) 50 sources

Figure 6.10: Cross traffic estimated using the reconstruction method versus actual traffic in the existence of 30 and 50 sources of cross traffic.



(a) 30 sources (b) 50 sources

Figure 6.11: Cross traffic estimated using the interpolation-based method (with $T = 0.05$) versus actual traffic in the existence of 30 and 50 sources.



(a) 30 sources (b) 50 sources

Figure 6.12: Cross traffic estimated using the Interpolation-based method (with $T = 0.5$) versus actual traffic in the cases that 30 and 50 sources of cross traffic are present.

# sources	30	40	50	60	70	80	90
Link util.	0.39	0.56	0.78	0.87	0.89	0.90	0.91
err	0.06	0.02	0.04	0.08	0.06	0.05	0.05
std	0.17	0.21	0.24	0.22	0.21	0.19	0.18

Table 6.4: Relative mean error and the standard deviation of errors under the prediction-based method.

# sources	30	40	50	60	70	80	90
Link util.	0.39	0.56	0.78	0.87	0.89	0.90	0.91
err	0.02	0.04	0.03	0.06	0.09	0.08	0.07
std	0.13	0.19	0.22	0.20	0.19	0.18	0.17

Table 6.5: Relative mean error and the standard deviation of errors under the reconstruction method.

the mean value of the cross traffic very well. On the other hand, the estimates obtained using the interpolation method oscillate around the actual values. This is because: For the prediction based method, the prediction result is the linear combination of the N past history information, or we can think of the result to be the weighted average of the N history values. From the frequency's point of view, the predictor is equivalent to a low pass filter, so the output of the predictor can keep the lower frequency band of the cross traffic, while the higher frequency band is filtered out.

For the reconstruction based method, the reconstructed process is guaranteed to have the same autocorrelation structure of the original process (asymptotically). Since the autocorrelation structure is the second order statistics, except Gaussian processes, the first and the second order statistics of a random process cannot uniquely determine a random process. Because the real cross traffic is not real Gaussian, although our reconstructed process has the same

# sources	30	40	50	60	70	80	90
Link util.	0.39	0.56	0.78	0.87	0.89	0.90	0.91
$err_{T=0.05}$	0.07	0.05	0.01	0.05	0.04	0.01	0.02
$std_{T=0.05}$	0.24	0.27	0.25	0.19	0.16	0.12	0.11
$err_{T=0.5}$	0.09	0.08	0.04	0.07	0.09	0.08	0.07
$std_{T=0.5}$	0.18	0.18	0.17	0.14	0.12	0.09	0.08

Table 6.6: Relative mean error and the standard deviation of errors under the interpolation-based method with $T = 0.05$ and $T = 0.5$.

first order (mean) and the second order (autocorrelation function) as the original process, it cannot catch the instantaneous variation of the original process.

But, for the the interpolation based method (especially when the sampling interval T is small, for example, $0.05s$), we interpolate one or two ($I_M = 1$ and $I_M = 2$) values in between two samplings, the interpolated value is heavily determined by the two samplings. While the intensive sampling can catch the instantaneously variation of the cross traffic, the interpolated value can also catch the the instantaneous information of the cross traffic. For larger T case, we observed that the interpolation method cannot trace the instantaneous variation of the cross traffic either.

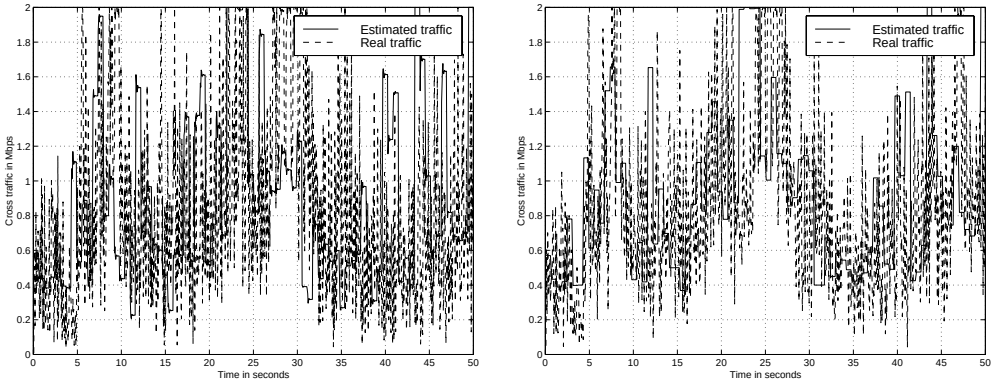
So, if we want to measure the mean value of the cross traffic, the first two methods suffice. If want to do short term traffic control, then the interpolation based method with smaller sampling interval can give good method to estimate the instantaneous change of the cross traffic.

- As shown in Figs. 6.34–6.12, the interpolation result oscillates much less significantly when T is large (e.g., $T = 0.5$) than when T is small ($T = 0.05$). This is because when T is large, the *FIR* filter filters out detailed information within the interval of length T and gives much smoother results.
- All three methods perform well under different utilizations in terms of relative mean error.

6.5.2 Experiment 2: Simulation with Real Internet Traces as Cross Traffic

In order to emulate the real network traffic, we downloaded real traffic traces from the ircache– [72]. We use the traces as the cross traffic in our simulation. The simulation scenario setup is the same as Section 6.5.1. For the interpolation method, we set $I_M = 2$. The simulation results are shown in Fig. 6.13 and Fig. 6.14.

The simulation results are consistent with the simulation results we obtained in Section 6.5.1, i.e., both prediction-base method and reconstruction-based method can catch the mean value of the cross traffic very well, while interpolation-base method catches instantaneous oscillation of the cross traffic. In this simulation, the measured relative estimation error is less than 0.1 for all the



(a) Prediction (b) Reconstruct

Figure 6.13: Cross traffic estimated using the prediction-based method and reconstruction-based method versus actual traffic in the existence of real Internet traces as sources of cross traffic.

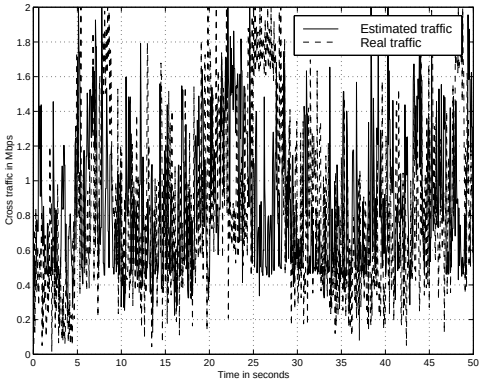


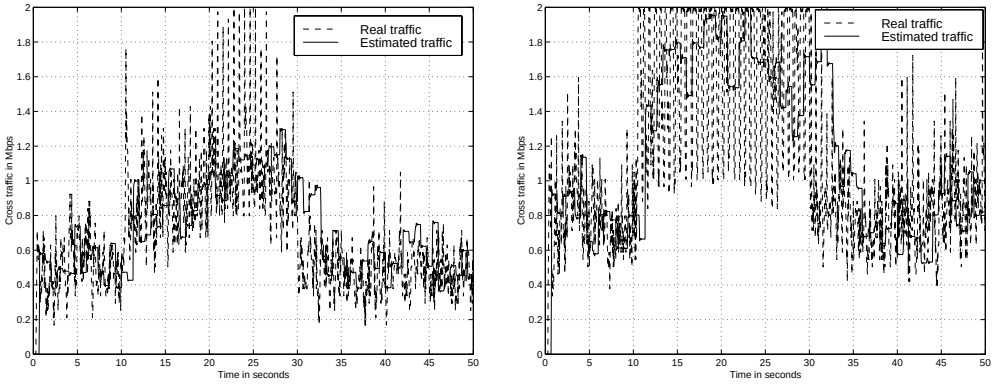
Figure 6.14: Cross traffic estimated using the interpolation-based method ($I_M = 2$) versus actual traffic in the existence of real Internet traces as sources of cross traffic.

three methods with the bottleneck link utilization equals 0.75. We also did simulation with higher bottleneck link utilization and get similar results.

Although using real traces as cross traffic in simulation cannot really emulate the situation in real network, the real traces do reflect the real situation of network to some extent. This experiment demonstrate the effectiveness of the three methods in a more realistic case.

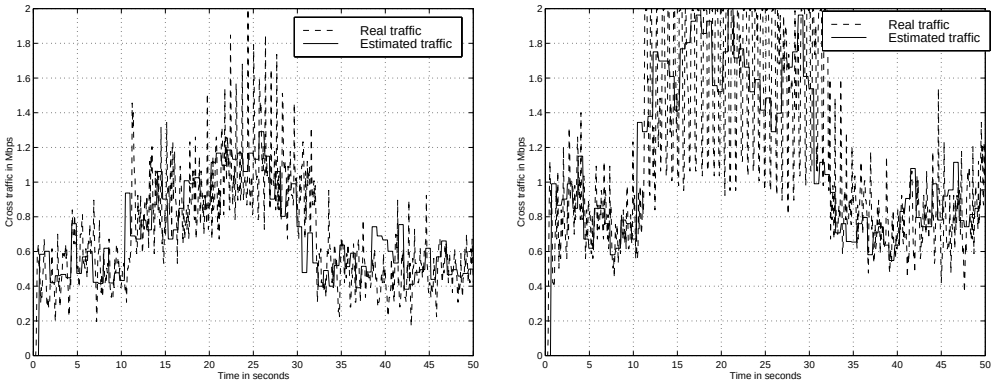
6.5.3 Experiment 3: Adaptability to Traffic Load Changes on the Bottleneck Link

Recall that In the first set of experiments all the connections that comprise the cross traffic last for 50 seconds, i.e., the amount of cross traffic does not change throughout the simulation. To



(a) 30 sources (b) 50 sources

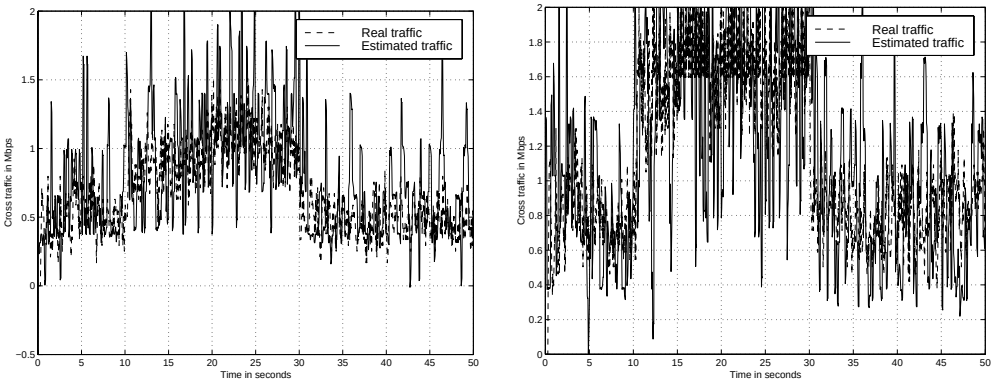
Figure 6.15: Cross traffic estimated using the prediction-based method versus actual traffic in the case that the amount of cross traffic dynamically changes.



(a) 30 sources (b) 50 sources

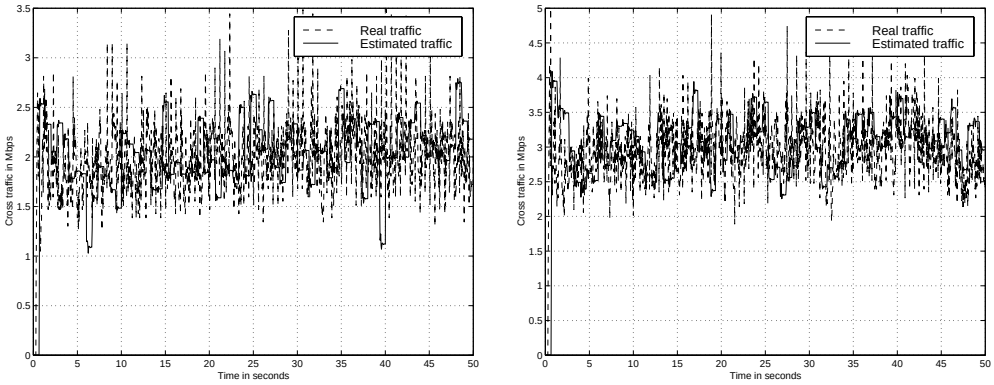
Figure 6.16: Cross traffic estimated using the reconstruction method versus actual traffic in the case that the amount of cross traffic dynamically changes.

evaluate the three methods in terms of their adaptability to the changes in the amount of cross traffic, we repeat the same experiments but vary the number of *effective* connections that comprise the cross traffic as follows. At time 0, $\frac{N}{2}$ cross-traffic connections commence, at time 10 s, another $\frac{N}{2}$ connections commence, and at time 30 s, $\frac{N}{2}$ connections terminate, where N varies from 30 to 100. All simulation runs last for 50 s. Figs. 6.15–6.17 give the corresponding simulation results. As shown in Figs. 6.15–6.17, all the three methods perform well and capture the changes in the amount of cross traffic almost immediately.



(a) 30 sources (b) 50 sources

Figure 6.17: Cross traffic estimated using the interpolation-based method (with $T = 0.05$ s) versus actual traffic in the case that the amount of cross traffic dynamically changes.



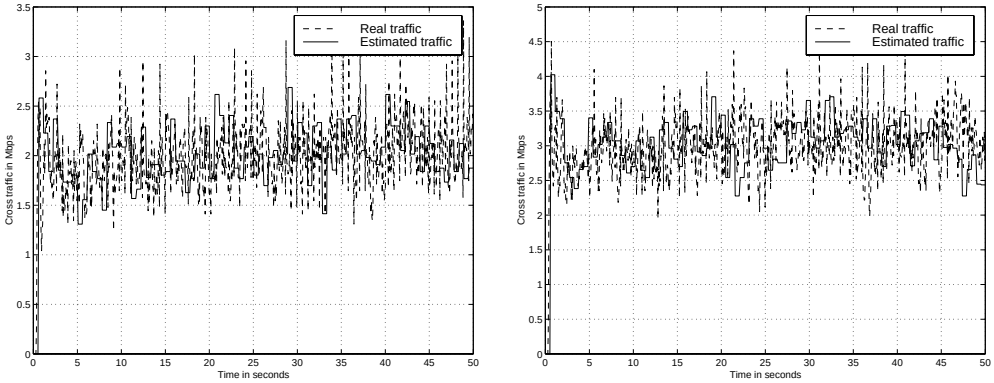
(a) Bandwidth = 4Mbps (b) Bandwidth = 6Mbps

Figure 6.18: Cross traffic estimated using the prediction-based method versus actual traffic in the cases that the bandwidth of the bottleneck link is 4Mbps and 6Mbps.

6.5.4 Experiment 4: Performance under Different Bottleneck Bandwidths

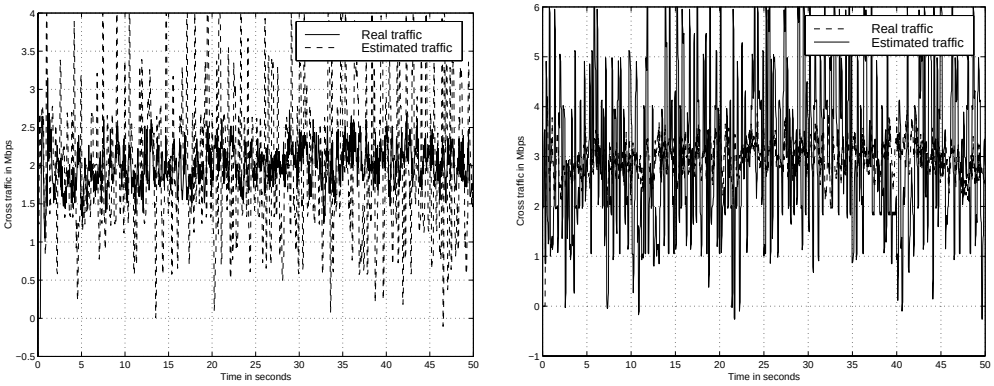
In this set of experiments, we repeat the same simulation as in Section 6.5.1 but vary the bandwidth of the bottleneck link from $2Mbps$ to $10Mbps$ and fix the number of cross traffic connections at 30.

Figs. 6.18–6.20 give the simulation results in the cases that the bandwidth of the bottleneck link is $4Mbps$ and $6Mbps$, respectively. Results similar to those in Section 6.5.1 are observed, indicating that under the same link utilization, the three methods are rather insensitive to the bandwidth of the bottleneck link. Table 6.7 gives *err* and *std* under the three methods.(For interpolation, $T = 0.05s$)



(a) Bandwidth = 4Mbps (b) Bandwidth = 6Mbps

Figure 6.19: Cross traffic estimated using the reconstruction method versus actual traffic in the cases that the bandwidth of the bottleneck link is 4Mbps and 6Mbps.



(a) Bandwidth = 4Mbps (b) Bandwidth = 6Mbps

Figure 6.20: Cross traffic estimated using the interpolation-based method (with $T = 0.05$ s) versus actual traffic in the cases that the bandwidth of the bottleneck link is 4Mbps and 6Mbps.

Bandwidth	2M	4M	6M	8M	10M
Prediction:*err*	0.06	0.02	0.02	0.03	0.04
Prediction:*std*	0.15	0.13	0.10	0.11	0.10
Reconstruction:*err*	0.02	0.02	0.03	0.04	0.05
Reconstruction:*std*	0.14	0.11	0.09	0.08	0.08
Interpolation:*err*	0.09	0.08	0.07	0.07	0.08
Interpolation:*std*	0.24	0.25	0.26	0.25	0.24

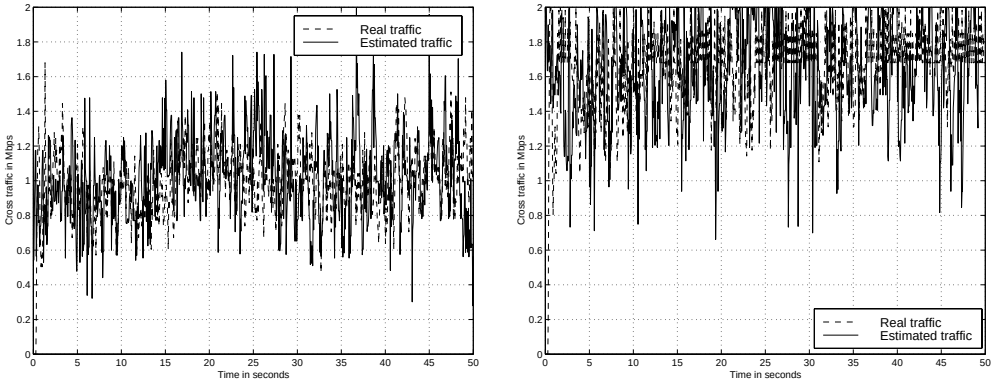
Table 6.7: The relative mean error and the standard deviation of errors under different bottleneck bandwidths.

6.5.5 Experiment 5: Effect of Varying the Number of Interpolated Values on Performance

In this set of experiments, we study the effect of varying the number, I_M, of interpolated values between two samples in the interpolation-based method. We repeat the same experiments as in Sections 6.5.1–6.5.3 but use the interpolation-based method with $I_M = 2$ for inferring the amount of cross traffic (set $T = 0.05s$). Figs. 6.21–6.22 give the corresponding simulation results. It turns out that the Interpolation-based method with $I_M = 2$ can capture not only the real mean value, but also the instantaneous value very well. Also, it incurs smaller values of *err* and *std*. To better illustrate this, we depict *err* and *std* under the prediction-based method, the reconstruction method, and the interpolation-based methods with $I_M = 1$ and $I_M = 2$ in Fig. 6.23. From the figure we can draw a conclusion: if we want to keep track of both the mean and the instantaneous value of the cross traffic while achieve smaller *err* and *std*, interpolation with $I_M = 2$,$T = 0.05s$ is the choice.

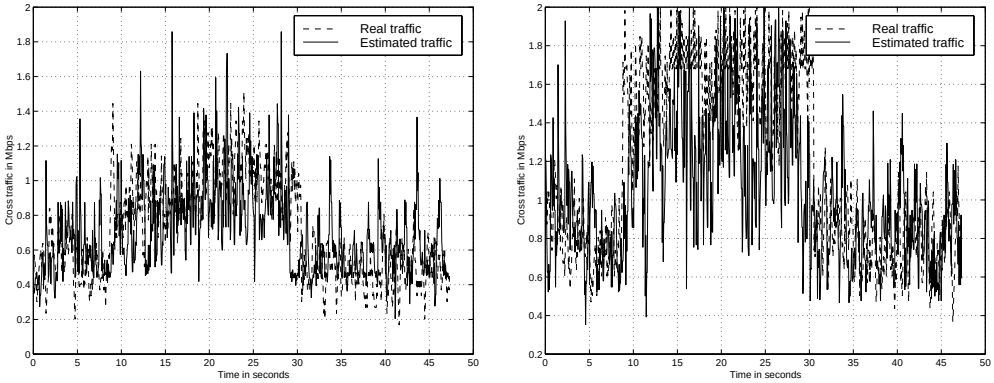
6.5.6 Experiment 6: Performance with Respect to Robustness

As mentioned in Section 6.1.1, we assume that probe packets only experience queuing at the bottleneck link. In real networks, this may not be true. In this set of experiments, we study the robustness of the three methods by relaxing the first two assumptions ((**A1**) and (**A2**)) in Section 6.1.1. The simulation scenario is depicted in Fig. 6.24: link $R1 - R2$ is the bottleneck link, but probe packets may also experience queuing at links $S - R1$ and $R2 - D$. The bandwidths of the links are, respectively, $B_{R1-R2} = 2Mbps$, $B_{S-R1} = 5Mbps$, $B_{R2-D} = 3Mbps$, and $5Mbps$ for all the other links. Cross traffic is generated by the left-hand-side hosts (labeled as senders), and



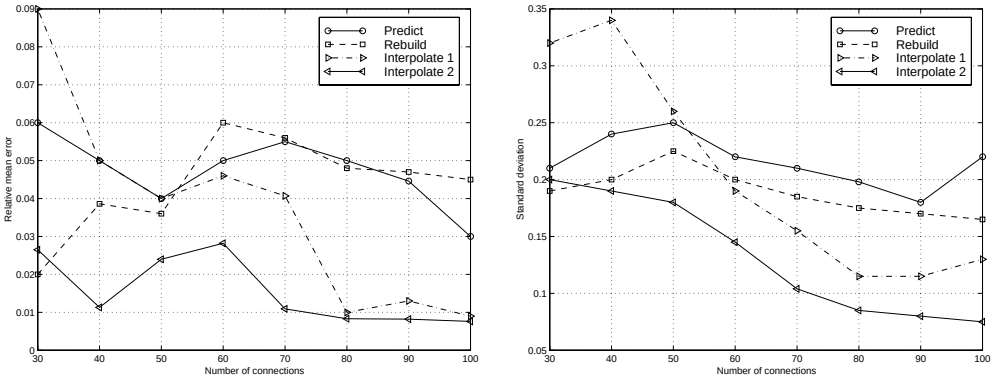
(a) 30 sources (b) 50 sources

Figure 6.21: Cross traffic estimated using the interpolation-based method (with $T = 0.05$ and $I_M = 2$ versus actual traffic in the existence of 30 and 50 traffic sources.



(a) 30 sources (b) 50 sources

Figure 6.22: Cross traffic estimated using the interpolation-based method (with $T = 0.05$ s and $I_M = 2$) versus actual traffic in the case that the amount of cross traffic dynamically changes.



(a) Relative mean error (b) Std dev of error

Figure 6.23: The relative mean error and standard deviation of the error under the prediction-based method, the reconstruction method, and the interpolation-based methods with $I_M = 1$ and $I_M = 2$.

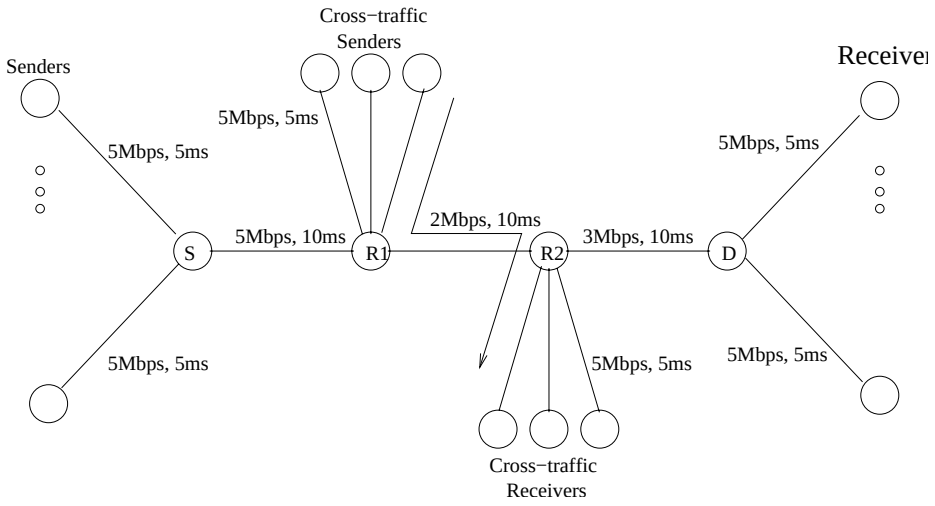


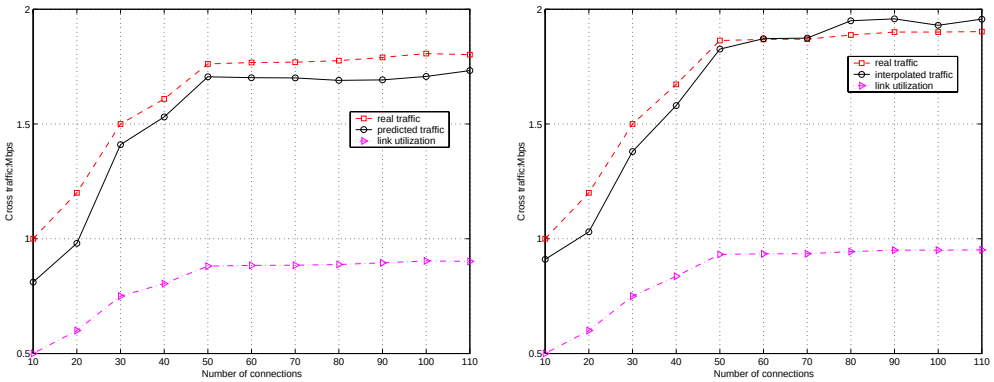
Figure 6.24: The topology with multiple bottleneck links.

traverses links $S - R1$, $R1 - R2$, and $R2 - D$. In addition, another additional 2-10 cross traffic connections are generated by hosts attached to $R1$ (labeled as cross-traffic senders) and traverse link $R1 - R2$.

As probe packets may get queued before or after link $R1 - R2$, the inter-arrival time τ_d between two back-to-back probe packets at the destination will be stretched/squeezed so that Eq. (6.3) does not hold. Specifically, if the second probe packet is queued before it arrives at link $R1 - R2$, when it arrives at link $R1 - R2$ the first probe packet may have already left. As a result, τ_d may be stretched, and the three methods will underestimate the amount of cross traffic that traverse link $R1 - R2$. Conversely, if the first probe packet is queued after it leaves link $R1 - R2$, τ_d may be squeezed, and the three methods will overestimate the amount of cross traffic.

Table 6.8 gives the mean error *err* and *std* under the three methods in the case that probe packets may be queued before/after the bottleneck link. As compared to Tables 6.4–6.6, the relative mean error in this scenario is only slightly larger (about 5%) than that in the idealistic case, while *std* is almost the same under both cases. This suggests that the three methods are pretty robust in the case that (**A2**) does not hold. In the figure to be presented below, we will see that the increase in *err* results from the effect of underestimating or overestimating the amount of cross traffic on the bottleneck link.

Fig. 6.25 gives the estimated and real mean values of cross traffic under different link utilizations.



(a) Prediction & Reconst. (b) Interpolation-based

Figure 6.25: Estimated and real mean values of cross traffic under the prediction-based, reconstruction, and interpolation-based methods.

# sources	30	40	50	60	70	80	90
link util.	0.41	0.62	0.82	0.88	0.92	0.93	0.93
Pred.:*err*	0.12	0.08	0.08	0.07	0.09	0.08	0.09
Pred.:*std*	0.19	0.23	0.20	0.21	0.21	0.20	0.18
Recon.:*err*	0.08	0.10	0.09	0.08	0.07	0.09	0.10
Recon.:*std*	0.14	0.18	0.22	0.18	0.19	0.17	0.18
Inter.:*err*	0.15	0.10	0.09	0.12	0.08	0.12	0.13
Inter.:*std*	0.21	0.23	0.20	0.17	0.15	0.14	0.11

Table 6.8: Estimation Mean Error *err* and *std* for Different Methods

As shown in Fig. 6.25, the estimated mean value of cross traffic is smaller than the real traffic under the prediction-based and reconstruction methods. In the interpolation-based method, when the link utilization is very high (≥ 0.95), the estimated mean value of cross traffic is higher than that of real traffic.

6.6 Empirical Study of the End-to-end Measurement Methods

In this section, we implement the three methods at both user level and kernel level. We first compare the performance of these two implementations, and our finding shows that the user level implementation tends to give a little bit overestimation of the mean value while the kernel level usually underestimates the real mean value. Although this difference exists, both implementations give relatively accurate estimate of the mean value of the cross traffic (with the relative error within 10%). Then, based on the user level implementation, we carry out real Internet experiments and

present the results in Section 6.6.2 and 6.6.3.

6.6.1 Kernel versus User Level Implementations

Implementation of the Proposed Methods

To carry out empirical study, there are usually two paradigms in implementing the proposed methods. First, the methods can be implemented at user level outside OS kernel. Second, the methods can be fulfilled at OS kernel level.

For the user level implementation, the resulting prototype implementations of the three methods are called *predict*, *rebuild* and *interp*, which represent the method of prediction, reconstruction and interpolation respectively. Each implementation consists of two parts, sending part and receiving part. The sending part sends probing packets in some specific pattern and the receiving part is in charge of measuring the interarrival time of the probing packets and inferring the cross traffic. The obvious advantage of implementing the methods in user level is the simplicity, and usually user level *C/C++* programming is needed. But the drawback is that the packets' sending and receiving processes may be interfered with the OS's queuing and job scheduling systems and may deviate from the pattern pre-defined.

The kernel level implementation is a little bit more tricky. By implementing at kernel level, we mean the sending part of the probing packets skips the application, *inet socket*, *socket*, transport and IP layers, and the probing packets are sent directly to the device driver (Ethernet card). The arriving packets are captured at the IP level (in *ip_input.c*) and the interarrival time is recorded in the system log file (in */var/log/messages*). The estimation of the cross traffic is done offline.

In order to send the packets directly to the device, we implement a kernel module called *udpsend* under Linux-2.4.23 kernel. In the module the probing UDP packets are sent directly to the hardware driver as pre-scheduled events. The major problem is that we must build the UDP packets from scratch. A packet in the Linux kernel is stored in an *skbuff* data structure, which includes the payload, UPD(TCP) header and IP header. Therefore, in order to build the UDP packets from scratch, we must artificially fill all the corresponding fields of the IP header and UDP header for the UDP packet. The fabricated UPD packets are delivered directly to the device driver by using *dev->hard_start_ximt*. To periodically send the probing packets, we use the kernel task scheduler

tq_struct (defined in *linux/tqueue.h*). At the receiver side, we modified the *ip_input.c* to capture the probing packets and the timestamp of the received packets are logged in the system log file.

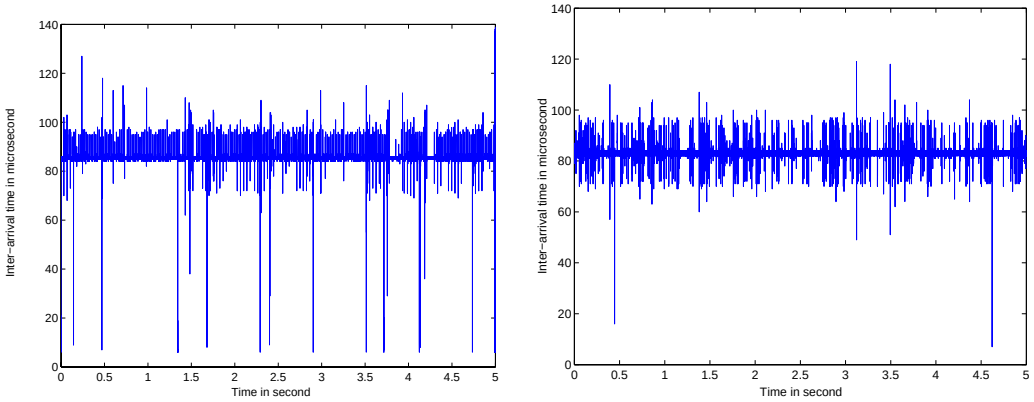
Since our methods assume that the bandwidth of the bottleneck link between two end hosts is known, we used a benchmark program *netperf* [69] to measure the approximate bottleneck link bandwidth between two end hosts. *netperf* attempts to transmit as many UDP (or TCP) packets as possible in a given time interval, and measures the maximum throughput. As the UDP stream of *netperf* competes against other traffic along the path and is non-responsive, the throughput serves as an approximation of the bottleneck link bandwidth between two end hosts.

Comparison of the Two Implementations

We carry out experiments to compare these two implementations. In the experiments, the two end hosts are stationed two hops away. Both end hosts are Del Precision 340 machine with single 2.2Ghz processor and 4Gbyte RAM. They are running Redhat Linux9.0 with kernel version 2.4.23 (the receiver's kernel has been modified to capture the probing packets). The *netperf* measurement shows that the maximum throughput between the sender and receiver is $95.5Mbps$.

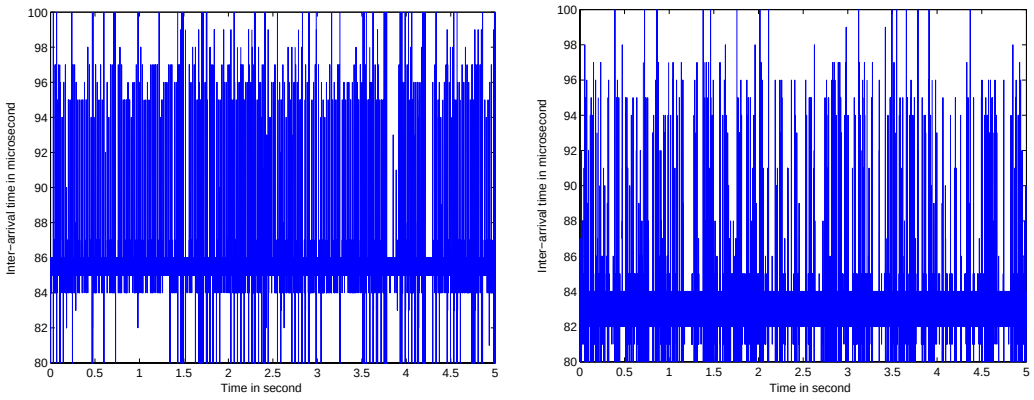
In the first experiment, we aim to check the difference of the interarrival time between two probing packets when there is not artificially generated cross traffic between the sender and receiver. At the sender, we keep on sending probing packets with interval less than 1000 byte/$95.5M \approx$ 83.7usec, where 1000 byte is the probing packet size. At the receiver we record the interarrival time between two probing packet and the results are shown in Fig. 6.26. We can see that, in both implementations, the interarrival time fluctuates around 80usec. In more details, we focus on the range of the interarrival time between 80usec and 100usec and show the results in Fig. 6.27. We can see that the interarrival time for the kernel module implementation is around 83usec, while that of user level implementation is around 85usec. The measured means are 83.4usec and 85.6usec, and the standard deviations are 3.3 and 5.1 for the kernel module and user level implementations, respectively.

Next, we artificially insert cross traffic between the sender and the receiver. The cross traffic are generated by constantly sending UDP packets from the sender to the receiver at an average rate of 12.5Mbps. The interarrival time of the probing packets for the two implementations are



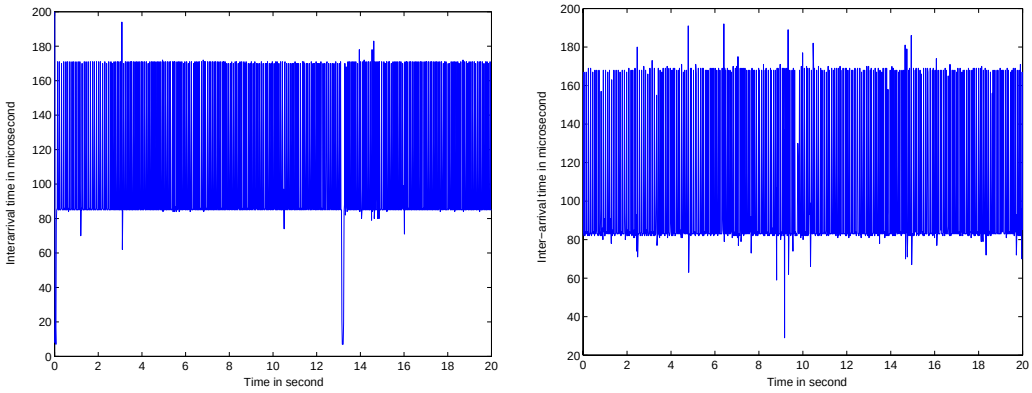
(a) User level (b) Kernel module

Figure 6.26: The interarrival time of the received packets.



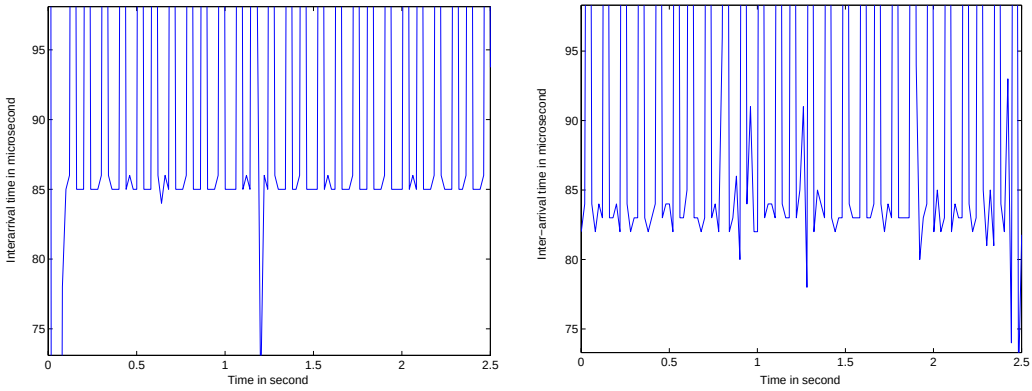
(a) User level (b) Kernel module

Figure 6.27: The interarrival time of the received packets between 80 and 100.



(a) User level (b) Kernel module

Figure 6.28: The interarrival time of the received packets with cross traffic.



(a) User level (b) Kernel module

Figure 6.29: The interarrival time of the received packets around 80 microsecond.

shown in Fig. 6.28. To see in more details, in Fig. 6.29 we show the detailed information around 85usec.

We can see that the interarrival time for the two implementations fluctuate around 100usec, but if we look in more details (Fig. 6.29), we find that for the kernel module implementation, in most cases the interarrival time is above 83usec, while that for the user level implementation is above 85usec. In average, the mean interarrival time of the probing packets for the kernel module implementation is about 2usec less than that of user level implementation. We observe similar results when we change the rate of the artificially generated cross traffic. With the interarrival time, by using Eq. (6.4), in Fig. 6.30 we show the real cross traffic and the estimated cross traffic under these two implementations.

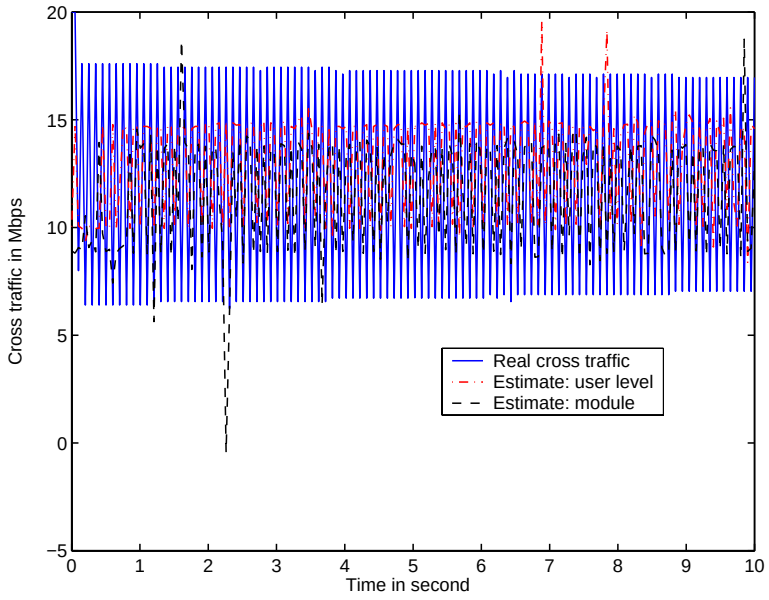


Figure 6.30: The real and measured cross traffic using both user level and kernel module implementations, when the mean cross traffic rate is 12.4Mbps.

Implementation	real mean(:Mbps)	12.5	24.07	35.6	44.3	52.7
User-level	measured mean	13.3	24.76	36.65	45.38	54.03
	measured std	3.2	4.5	4.7	5.9	6.3
Kernel-level	measured mean	11.8	23.60	34.71	43.41	51.53
	measured std	3.8	4.2	5.0	5.5	6.1

Table 6.9: The measured mean and std of the cross traffic.

We observe that under both implementations we can obtain relative good estimation. In terms of mean value, the real mean rate of cross traffic is 12.4 Mbps, and the mean rate of the measured cross traffic using user level implementation is 13.3 Mbps, while that of kernel module implementation is 11.8Mbps. So we can see that the user level implementation overestimates the real cross traffic while the kernel module implementation underestimates that. In both cases, the relative estimation error is less than 10%. We change the cross traffic rate and measure the cross traffic. In Fig. 6.31 we show the result when the mean rate of the cross traffic is 24.07Mbps. The measured mean cross traffic for user level implementation is 24.76Mbps and that of kernel module implementation is 23.60Mbps. Again, both of them give relatively good estimation of the cross traffic. We summarize our experimental results in Table 6.9.

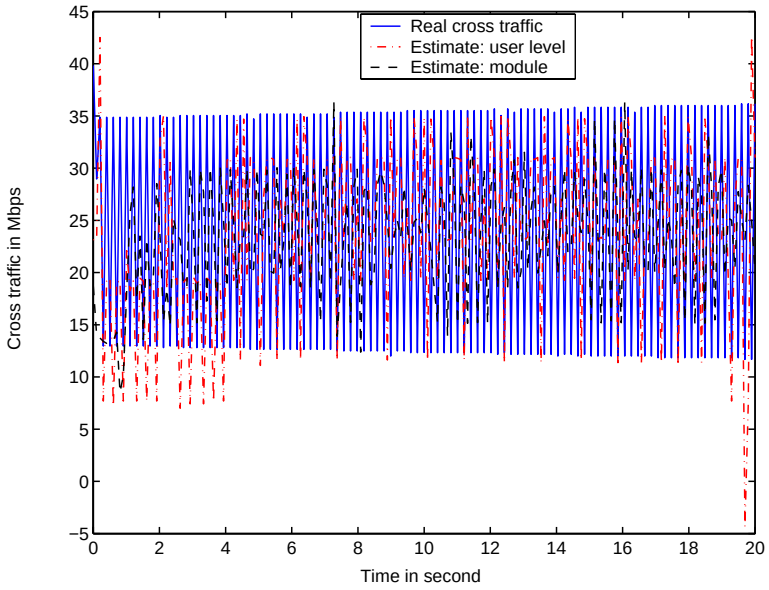
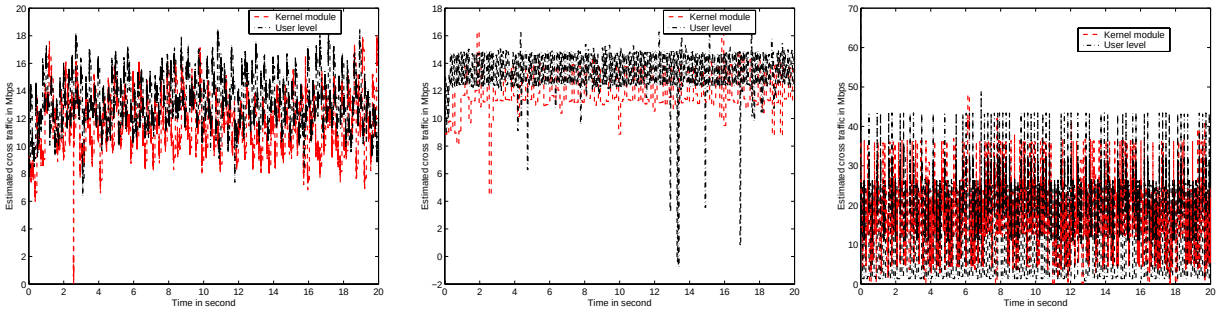


Figure 6.31: The real and measured cross traffic using both user level and kernel module implementations, when the mean rate of cross traffic is 24.07Mbps.

One thing worth mentioning is that, in the above experiments, the machines (sender and receiver) are lightly loaded (about 90% cpu time is occupied by the sender or receiver). To study the the impact of background processes on the interarrivals of probing packets, we also did experiments when the machines (sender, receiver or both) are heavily loaded. Specifically, we artificially generate some background processes and they occupied over 90% of the cpu time. We find that the interarrivals of probing packets do not change much compared with the lightly loaded case. The reason is that, according to [15], in doing process scheduling, the cpu time is divided into epochs and every process has a specified time quantum calculated at the beginning of each epoch. The base time quantum equals 210ms , which is also the smallest unit that can be assigned to a process. Furthermore, in order to avoid frequent context switch among the on-going processes, the Kernel's job scheduling algorithm tries to keep the time quantum as long as possible as long as the kernel's response time is not degraded. Therefore, it is very unlikely that a context switch will happen between the sendings or receivings of two probe packets (whose interval is at the level of microsecond), and the measured interarrivals of probing packet are not influenced.



(a) Prediction (b) Reconstruction (c) Interpolation

Figure 6.32: The estimated cross traffic for the three proposed methods when the mean cross traffic rate is 12.5Mbps.

Performance of the Proposed Methods under the Two Implementations

Next, we study the performance of the three proposed methods: prediction, reconstruction and interpolation under the two implementations. The experiments setting is the same as above, instead we use the proposed methods to estimate the cross traffic. Again we artificially generate cross traffic between the sender and the receiver. The average sending rate is 12.5Mbps. The cross traffic is almost the same as in Fig. 6.30 and is omitted here. The probing packet patterns for the three methods are the same as in Section 6.5. In Fig. 6.32, we show the estimated cross traffic for the three methods when the real mean rate of the cross traffic is 12.5Mbps (for other different cross traffic rates, we got similar results). We can see that, under the two implementations, both prediction and reconstruction methods can capture the mean value of the cross traffic. Similar to the results in Fig. 6.30, the estimated result under user level implementation overestimates the real mean rate and that of kernel module implementation underestimates the real mean rate, while in both cases, the relative error is small (less than 10%). But the interpolation results under both implementations are not satisfactory. This is because the interpolation method heavily depends on the instantaneous estimation of the cross traffic, and due to the fluctuation of the interarrival time of the probing packets, the instantaneous estimation is not accurate enough to do interpolation.

In summary, we find that:

- The average interarrival time between two packets under kernel module implementation tends to be a little bit longer (around 2 microsecond) than that of user level implementation. This little difference causes mild difference of the estimates of the mean cross traffic, i.e., the result

from user-level implementation tends to underestimate the real mean value of the cross traffic, while that of kernel-level implementation overestimates the real mean value. Moreover, both implementations provides relatively accurate estimates of the real mean value (with relative error less than 10%).

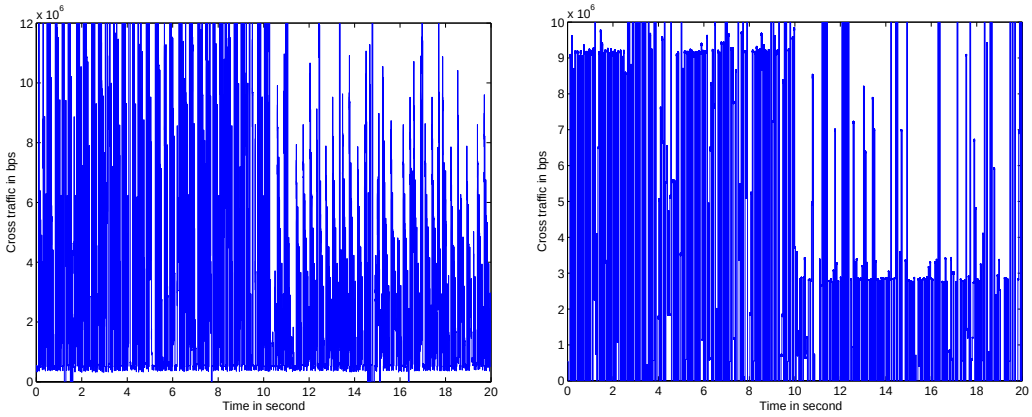
- The proposed methods under both implementations give similar estimates of the cross traffic. Specifically, the prediction and reconstruction methods provide good estimate of the mean value of the cross traffic, while the interpolation method fails to give satisfactory results. In the Section 6.6.4, we will get back to the problem and a minor revised interpolation method can remedy this problem.

In the following sections (experiments on both campus network and real Internet), we will focus on the user level implementation.

6.6.2 Empirical Results on a Campus Network

In this section, We measure the cross traffic between two hosts which are 3 hops away on a campus network. One host (the receiver) is a Dell Precision 340 machine with single 2.2 GHz processor and 4GByte RAM. It is running Redhat Linux 8.0 with kernel version 2.4.9-34, and is connected to the campus backbone through a 100Mbps Ethernet link. The other host (the sender) is a Sum Ultra 1 Model 140 machine with UltraSPARC 1 processor running at 140MHz and 64MByte RAM. It is running SunOS 5.8 and is connected to the campus backbone through a 10Mbps Ethernet link. To measure the cross traffic along the path, we first use *netperf* to get the bottleneck link bandwidth, which is 9.5Mbps.

In our implementations, we set the bottleneck link bandwidth to be 9.5Mbps. To show the adaptability of our methods, we artificially generate cross traffic between the two end hosts, which changes at some specific time point. Specifically, we artificially generate 20 seconds cross traffic between the two end hosts. During the first 10-second period, the cross traffic is transmitted at an average rate ranging from 2Mbps to 9Mbps, and during the second 10-second period the transmission rate reduces by half. Fig. 6.33(a) shows the measured cross traffic using prediction method when the cross traffic sending rate is 4Mbps. We can observe that the measured cross traffic oscillates around 4Mb/s during the first 10-second period and 2Mb/s during the second 10-second

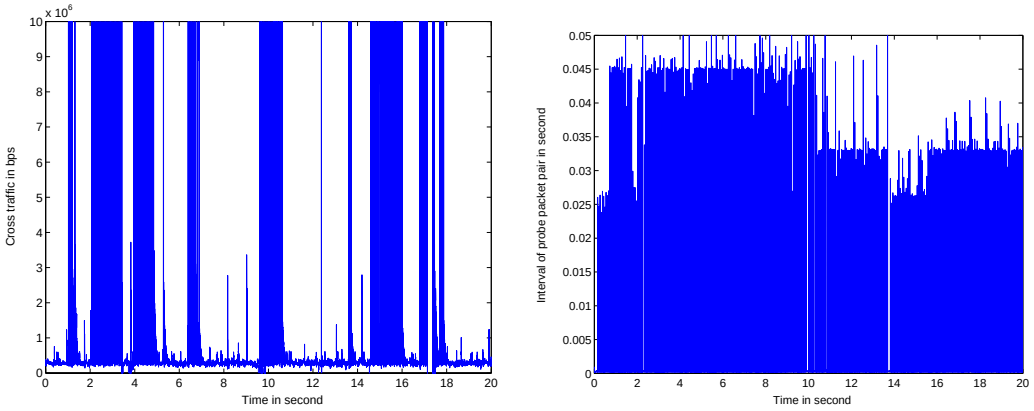


(a) Prediction (b) Reconstruction

Figure 6.33: Estimated values of cross traffic under the prediction-based and reconstruction methods.

period (for other sending rate we got similar results). And, the measured result can capture the instantaneous change of the cross traffic. The reconstruction method gives similar result which is shown in Fig. 6.33(b).

For the interpolation based method, we set $I_M = 2$ and $T = 0.02$ (we got similar results for other I_Ms and Ts), the measured cross traffic is shown in Fig. 6.34 (a). The result is not encouraging. The reason is due to the fact that the interpolation based method depends heavily on the instantaneously sampled cross traffic, which is further dependent on the accuracy of the interval of probe packet pairs. Since the interval of probe packet pair changes dramatically, the instantaneously sampled cross traffic could be far from the real value. To say this point, we measured the interval of the received probe packet pairs and show the result in Fig. 6.34 (b). We can see that the intervals of the received probe packet pairs range from 0 to 0.05 second, and this explains why the interpolation method fails to give good estimate of the cross traffic. To find why the intervals of the received probe packet pairs change dramatically, we traced the sending process down to the physical device right before the packets are put into the physical link. In particular, we record the physical link starting transmission time of packets — *trans_start* in */drivers/net/eepro100.c* and find that the interval between the starting transmission time of the probe packets oscillates dramatically. The major reason, if not the unique one, of this oscillation can be attributed to the "noise" inherent in the hardware interrupt system and the MAC layer CSMA/CD protocol. Since, according to [113]



(a) Interpolation result (b) Interval of packet pair

Figure 6.34: measured values of cross traffic under the interpolation base methods and the real intervals of probe packet pairs.

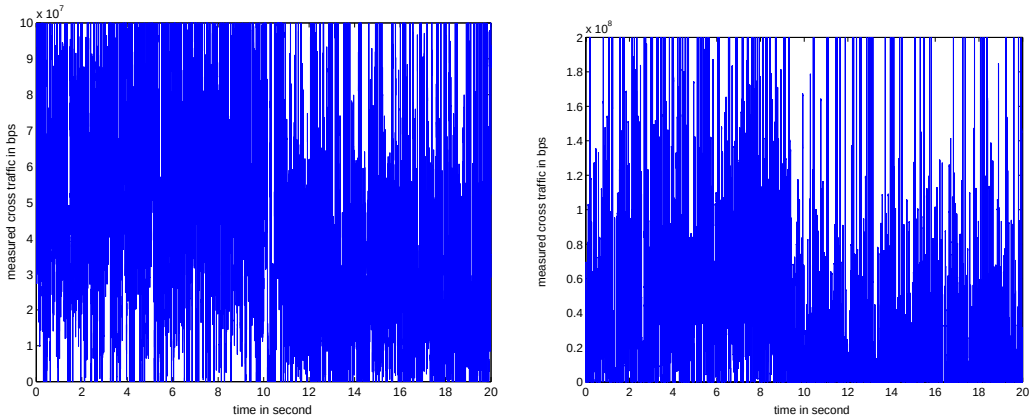
(chapter 14), hardwares can forget what they are doing and the system can lose an interrupt, and this kind of problem is common with hardware devices. The device drivers deal with this kind of problem by setting timers and resume the transmission when the timers expire. Furthermore, the media contention scheme also introduces oscillation to the interval of start transmission time. On the receiver side, the packet receiving from the physical link is also done by the hardware interrupt system and similar things can happen. All these factors result in the scene we observed in Fig. 6.34 (b).

In Section 6.6.4 we modify the interpolation method to amortize the adverse effect of this oscillation so that relatively good results can be obtained.

6.6.3 Empirical Results on the Internet

We next carry out the measurement experiments in real Internet. Using *traceroute*, we know that the probe packets traverse 13 hops along the path of UIUC → RICE. Again, we use *netperf* to measure the bottleneck link bandwidth, which is 93.4Mbps.

In the experiments, we also artificially generate cross traffic with average sending rate changing from 20Mbps to 80 Mbps. The cross traffic lasts for 20s, and for the second half period the sending rate reduces by half. The results are shown in Fig. 6.35 for the prediction and reconstruction based method.



(a) Prediction (b) Reconstruction

Figure 6.35: Estimated values of cross traffic under the prediction-based and reconstruction.

Fig. 6.35 shows the result for prediction (reconstruction) when the generated cross traffic is transmitted at rate 40Mbps (similar results are obtained for other sending rate). We can observe that the two methods can give relatively good measurement of the cross traffic and can adapt to the dynamic change of the cross traffic promptly. For the interpolation based method, we observe similar unsatisfactory result as Section 6.6.2 and the result is shown in Fig. 6.36. In order to overcome the drawback of the interpolation based method, next, we consider a minor revision of the interpolation based method.

6.6.4 A Revision of the Interpolation Based Method

To overcome the interpolation based method's excessive dependence on the instantaneous sampling value of the cross traffic, a straight forward remedy is to make $m > 1$ samples between time interval T (Section 6.4.2) instead of making only 1 sample. We use the average value of the m samples to interpolate the values during T. In our empirical study, we set $5 \leq m \leq 10$ and the results (when $m = 8$) are shown in Fig. 6.37, and the experimental settings for the campus and the Internet experiments are the same as those in Section 6.6.2 and Section 6.6.3 respectively. We can observe that, the improvement of the revised method is obvious.

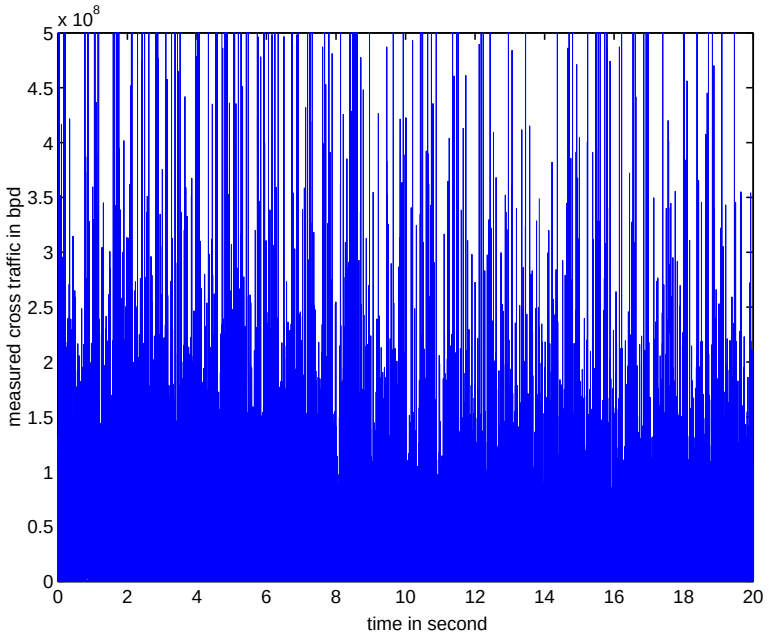
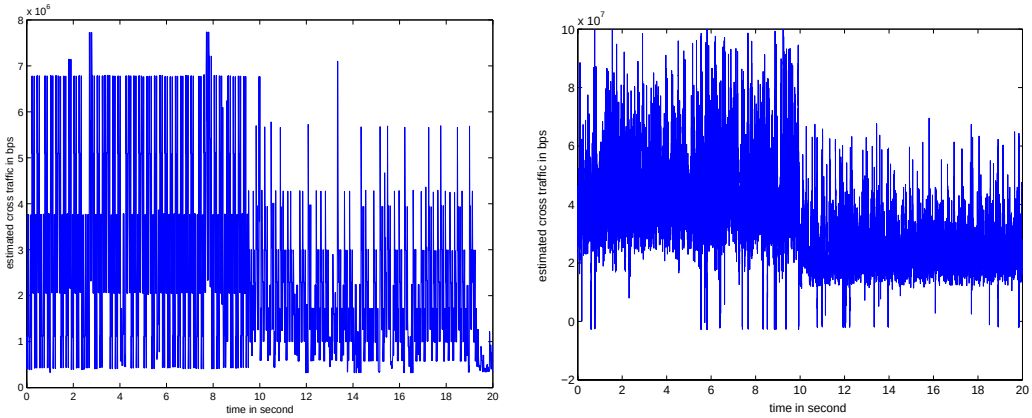


Figure 6.36: The measured cross traffic under the interpolation based method.



(a) Campus experiment (b) Internet experiment

Figure 6.37: Estimated values of cross traffic under the revised interpolation based method.

6.7 Summary

In this chapter, we demonstrate that the self similarity property of cross-traffic can be exploited to infer on an end-to-end basis the amount of cross traffic on the bottleneck link, and investigate three such theoretically-grounded methods: prediction, reconstruction, and interpolation. The simulation study indicates that the prediction-based and reconstruction methods can give good mean measurement of cross traffic, while the interpolation method can, with proper design of the *FIR* filter, capture both the mean and instantaneous values of cross-traffic. All three methods are adaptive to the dynamic change of cross traffic and are quite robust in the presence of multiple bottleneck links on an end-to-end path. The empirical study (for both Campus network and Internet) shows that the prediction and reconstruction based methods can catch the mean value of the cross traffic well. But the interpolation based method fails at this point. The knot of the problem with the interpolation based method lies in the fact that the interpolation method depends excessively on the instantaneously obtained measurement of the cross traffic and that measurement usually fluctuates dramatically. By noticing this, a minor revision of the interpolation method provides much better results.

Chapter 7

Biased Systematic Sampling

Internet traffic passive measurement techniques—sampling techniques are very important to understand the traffic characteristics of the Internet [39, 48], and have received increasing attention. In this chapter, we perform an in-depth, analytical study of three sampling techniques for self-similar Internet traffic, namely static systematic sampling, stratified random sampling and simple random sampling. We show that while all three sampling techniques can accurately capture the Hurst parameter (second order statistics) of Internet traffic, they fail to capture the mean (first order statistics) faithfully, due to the bursty nature of Internet traffic. We also show that static systematic sampling renders the smallest variation of sampling results in different instances of sampling (i.e., it gives sampling results of high fidelity). Based on an important observation, we then devise a new variation of static systematic sampling, called *biased systematic sampling (BSS)*, that gives much more accurate estimates of the mean, while keeping the sampling overhead low. Both the analysis on the three sampling techniques and the evaluation of BSS are performed on synthetic and real Internet traffic traces. The performance evaluation shows that BSS gives a performance improvement of 40% and 20% (in terms of efficiency) as compared to static systematic and simple random sampling.

Note that the traffic process $X(t)$ considered in the chapter is rather general, and can be either an individual OD-flow or the aggregate of several/all OD-flows that traverse a router. After $X(t)$ is specified, the proposed sampling technique can be used to, for example, estimate the mean of the aggregate traffic of several (selected) OD flows between the west and east coasts in the States. Note also that although it is feasible to log each and every individual packet and record the entire flow time series $X(t)$, the process of collecting such an enormous amount of samples can only

be carried out at a small number of ISP routers that are equipped with DAG packet collection cards and large memory. The large amount of data is then analyzed off-line to better understand the traffic characteristics. Sampling remains as an effective and economical technique to on-line collect/estimate parameters that characterize the traffic.

The rest of the chapter is organized as follows. After introducing the three traditional sampling techniques in Section 7.1, we investigate analytically in Section 7.2 whether or not the three sampling techniques accurately capture the Hurst parameter of the process to be measured and provide a *SNC* that a sampling strategy must satisfy in order to keep the second order statistics (and hence Hurst parameter). Then, we compare in Section 7.3 the average variance of the sampling results obtained by the three techniques. Following that in Section 7.4, we demonstrate with both synthesized and real Internet traces that all three techniques fail to capture the real mean of Internet traffic and present BSS in detail. Finally we present our performance study (again based on both synthesized and real traces) in Section 7.5. The chapter concludes with Section 7.6.

7.1 Traffic Process and Three Traditional Sampling Techniques

In this section, we define the traffic process $X(t)$ and introduce the three commonly used sampling techniques which set the stage for subsequent derivation and discussion.

A general definition of the traffic process Let $\{X(t), t \in Z+\}$ be a time series which represents the traffic process measured at some fixed time granularity. As we have mentioned, the traffic process can be individual OD-flow or the aggregation of several OD-flows or any other flows the researchers are interested in. To make our approach a generic one, our definition on $X(t)$ is rather general.

Three sampling techniques Generally speaking, the larger the sampling set, the more accurately the original process can be characterized. The price one has to pay is, however, the more CPU processing time and buffer space. Indeed there exists a trade-off between the sampling rate and the accuracy of sampling results. Three categories of sampling techniques have been commonly used in measuring Internet traffic: systematic sampling, stratified random sampling, and simple random sampling (Figure 7.1). In systematic sampling, every C_tth element (e.g., packet or time

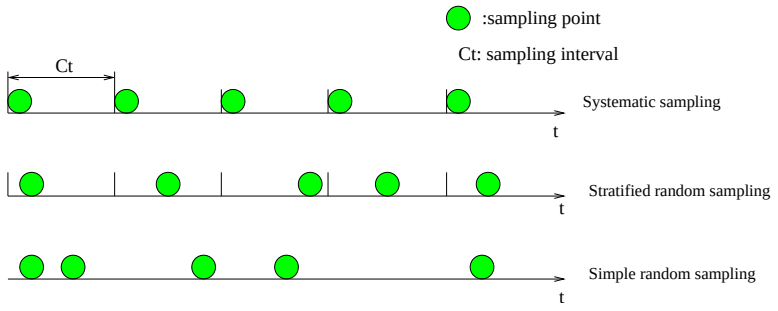


Figure 7.1: An illustration of the three sampling techniques.

instance) of the parent process is deterministically selected for sampling, starting from some starting sampling point. In stratified random sampling, the time axis is divided into intervals of length C_t, and one sample is randomly selected in each interval. In simple random sampling, N packets are randomly selected from the entire population.

7.2 Hurst Parameter of the Sampled Process

In this section, we first investigate whether or not the three sampling techniques accurately capture the Hurst parameter of Internet traffic. This is done by deriving the autocorrelation function of the sampled process obtained from the three sampling techniques. (Note that we do not intend to devise a procedure to estimate the Hurst parameter, but instead derive the Hurst parameter (through calculation of the autocorrelation function) of the sampled process and compare it with that of the original process.) Then we derive a *SNC* that a sampling technique has to satisfy in order to retain the autocorrelation structure of the original process.

7.2.1 Systematic Sampling

Let $X(t)$ and $g(t)$ denote the original and sampled process, and H_x and H_g the Hurst parameter of $X(t)$ and $g(t)$ respectively. Without loss of generality, t is discretized to be integer numbers: $0, 1, 2, 3....$ For systematic sampling, let C_t be the sampling interval. Then we have[1]

$$g(t) = X(C_t t), t = 0, 1, 2, ... \tag{7.1}$$

[1]Without loss of generality, we denote the starting point of systematic sampling to be $t = 0$.

Let $R_X(\tau)$ and $R_g(\tau)$ denote the autocorrelation function of $X(t)$ and $g(t)$, and $F(t)$ and $G(t)$ denote the CDF of $X(t)$ and $g(t)$ respectively. Then we have

$$R_g(\tau) = E(g(t)g(t-\tau)) = E(X(C_t t)X(C_t t - C_t\tau)) = \int X(C_t t)X(C_t t - C_t\tau)dF(t). \tag{7.2}$$

Let $C_t t = u$. Then Eq. (7.2) can be re-written as

$$R_g(\tau) = \int X(u)X(u-\tau)C_t^{-1}dF(t) = C_t^{-1} \cdot R_X(\tau). \tag{7.3}$$

Hence $R_g(\tau) = C_t^{-1}R_X(\tau) \sim A\tau^{-\beta}$ as $\tau \to \infty$, where A is a constant. Also, we have $H_g = H_x = \frac{2-\beta}{2}$, where $0 < \beta < 1$. The above derivation implies that the sampled process obtained by the static systematic sampling technique has the same Hurst parameter as the original process.

7.2.2 Stratified Random Sampling

Recall that in stratified random sampling, the time axis is divided into interval of length C_t, and one sample is randomly selected in each interval. Using the same notation as in Section 7.2.1, we have

$$R_g(\tau) = E(g(t)g(t-\tau)) = E(X(C_t t + \tau_1)X(C_t t - C_t\tau + \tau_2)),$$

where τ_1 and τ_2 are random variables that represent the time lags after the beginning of the tth and $(t-\tau)$th bucket respectively. $R_g(\tau)$ can be further written as

$$\begin{aligned} R_g(\tau) &= E(E(X(C_t t + \tau_1)X(C_t t - C_t\tau + \tau_2)|\tau_1, \tau_2)) \\ &= E(C_t^{-H-1}R_X(\tau + \frac{\tau_1 - \tau_2}{C_t})) \\ &= E(C_t^{-H-1}R_X(\tau + \tau')), \end{aligned}$$

where $\tau' = \frac{\tau_1 - \tau_2}{C_t}$.

By Eq. (2.1), we have

$$R_g(\tau) \sim E(D \cdot (\tau + \tau')^{-\beta}) = \int D \cdot (\tau + \tau')^{-\beta} f_{\tau'} d\tau',$$

where D is a constant related to C_t, and $f_{\tau'}$ is the probability density function (pdf) of τ'. As both τ_1 and τ_2 are uniformly distributed in $[0, C_t]$, we have

$$f_{\tau'}(x) = \begin{cases} 1+x, & \text{if } -1 \le x \le 0, \\ 1-x, & \text{if } 0 \le x \le 1, \end{cases} \tag{7.4}$$

and hence

$$\begin{aligned} R_g(\tau) &\sim \int_{-1}^{1} D \cdot (\tau + \tau')^{-\beta} f_{\tau'} d\tau' \\ &\sim D\tau^{-\beta} \int_{-1}^{1} (1 - \beta \frac{\tau'}{\tau}) f_{\tau'} d\tau' \\ &= D \cdot \tau^{-\beta} \text{as } \tau \to \infty. \end{aligned} \tag{7.5}$$

The last equality results from the fact that $E(\tau') = 0$. By Eq. (7.5), we conclude that the sampled process obtained by the stratified random sampling technique has the same Hurst parameter as the original process.

7.2.3 Simple Random Sampling

In simple random sampling, N samples are randomly selected from the entire population of M samples. That is, with probability $\rho = N/M$ a sample is selected. Let t_0 denote the sampling point in $X(t)$ corresponding to the tth sample $g(t)$. Then we have

$$\begin{aligned} R_g(\tau) &= E(g(t)g(t+\tau)) \\ &= E(X(t_0)x(t_0+a)) \;=\; R_X(a), \end{aligned}$$

where $a \ge \tau$ is a random variable. Since

$$\Pr(a = \tau + i) = \begin{pmatrix} \tau + i - 1 \\ i \end{pmatrix} \rho^{\tau} (1-\rho)^i, i = 0, 1, 2.., \tag{7.6}$$

we have

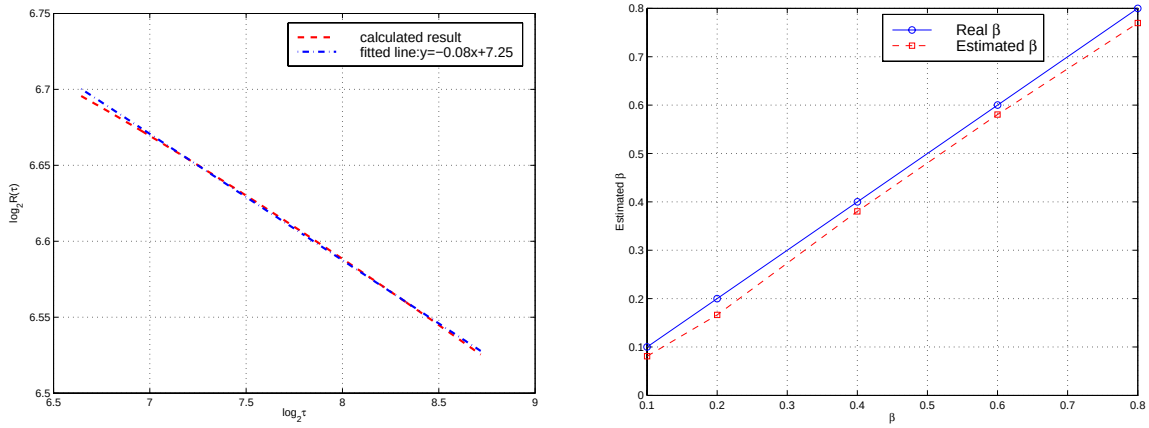
$$\begin{aligned} R_g(\tau) &= \sum_{a=\tau}^{\infty} R_X(a) \cdot \Pr(a) \\ &= \sum_{i=0}^{\infty} R_X(\tau+i) \begin{pmatrix} \tau+i-1 \\ i \end{pmatrix} \rho^{\tau}(1-\rho)^{i} \\ &\sim \sum_{a=\tau}^{\infty} \Gamma a^{-\beta} \begin{pmatrix} a-1 \\ a-\tau \end{pmatrix} \rho^{\tau}(1-\rho)^{a-\tau} \\ &= \sum_{a=\tau}^{\infty} \Gamma a^{-\beta} \frac{(a-1)!}{(a-\tau)!(\tau-1)!} \rho^{\tau}(1-\rho)^{a-\tau}, \end{aligned} \tag{7.7}$$

where Γ is a constant. Using the *Sterling* equation, we can further approximate Eq. (7.7) as

$$\begin{aligned} R_g(\tau) &\approx \frac{\Gamma\rho^{\tau}}{\sqrt{2\pi(\tau-1)}(\tau-1)^{\tau-1}e^{-(\tau-1)}} \sum_{a=\tau}^{\infty} \frac{a^{-\beta}(a-1)^{a-1/2}e^{-(a-1)}}{(a-\tau)^{a-\tau+1/2}e^{-(a-\tau)}} \cdot (1-\rho)^{a-\tau} \\ &= \frac{\Gamma\rho^{\tau}(1-\rho)^{-\tau}}{\sqrt{2\pi}(\tau-1)^{\tau-1/2}} \sum_{a=\tau}^{\infty} \frac{a^{-\beta}(a-1)^{a-1/2}(1-\rho)^{a}}{(a-\tau)^{a-\tau+1/2}} \\ &\triangleq \hat{\Gamma} \sum_{a=\tau}^{\infty} \frac{a^{-\beta}(a-1)^{a-1/2}(1-\rho)^{a}}{(a-\tau)^{a-\tau+1/2}}, \end{aligned} \tag{7.8}$$

where $\hat{\Gamma} = \frac{\Gamma(\frac{\rho}{1-\rho})^{\tau}}{\sqrt{2\pi}(\tau-1)^{\tau-1/2}}$.

Since no closed form result can be obtained from Eq. (7.8), We use Matlab to find the relation between $R_g(\tau)$ and τ. Specifically, We fit the value of $R_g(\tau)$ (calculated from Eq. (7.8)) to $const \cdot \tau^{\hat{\beta}}$ and depict the estimated value $\hat{\beta}$ and the real value of β in Fig. 7.2. In Fig. 7.2 (a) we fit the calculated result of $R_g(\tau)$ (after taking $\log_2$ on both τ and $R(\tau)$) to a line with slope $\hat{\beta} = -0.08$, where the real value is $\beta = 0.1$. By changing the real value of β from 0.1 to 0.8, we perform the same operation and report the estimated value of $\hat{\beta}$ in Fig. 7.2 (b). As shown in Fig. 7.2 (b), the values of $\hat{\beta}$ and β agree very well and hence $H_g \approx H_x$. Note the small gap between the values of $\hat{\beta}$ and β is due to the truncation error on the right hand side of Eq. (7.8), i.e., in calculating $R_g(\tau)$, we cannot sum up an infinite number of terms (from $a = \tau$ to ∞) and have to approximate the right hand side of Eq. (7.8) with a finite number of terms.



(a) The calculated result of $R_g(\tau)$ is fit into the line $log_2 R_g(\tau) = -0.08 log_2 \tau + 7.25$; the real value of β is 0.1.

(b) $\hat{\beta}$ versus β.

Figure 7.2: Estimated and actual values of β.

7.2.4 Sufficient and Necessary Condition for Accurately Capturing the Hurst Parameter

In Section 7.2.1–7.2.3, we have shown that the sampled process generated by all three sampling techniques has the same Hurst parameter as the original process. A more general question is then: given a sampling technique, how do we check if the sampled process generated by this technique has the same Hurst parameter as the original process? To answer the question, we derive a sufficient and necessary condition (SNC) which a sampling technique has to satisfy in order to preserve the same second order statistics (and therefore Hurst parameter) in the thinned process.

We generalize the sampled process generated by a sampling method to be a point process $Z_n, n = 1, 2, 3...$, which represents the series of sampling points. The intervals between any two consecutive sampling points are defined as $T_i = Z_{i+1} - Z_i, i = 1, 2, ...$ T_i's are i.i.d random variables with the probability density function $h_s(x)$ for the continuous case and the probability mass function $H_s(x)$ for the discrete case. Note that Z_n is a renewal process with the renewal interval distribution h_s or H_s. A sampling method (and hence the sampled process generated by the sampling method) is generated by h_s or H_s. For example, the function H_s for systematic sampling is $H_s(C_t) = \Pr(T_i = C_t) = 1$ and $H_s(x) = 0$ for $x \neq C_t$, while the function h_s for stratified

random sampling method is

$$h_s(x) = \begin{cases} \frac{1}{C_t^2}x, & \text{if } 0 \le x \le C_t, \\ \frac{2}{C_t} - \frac{1}{C_t^2}x, & \text{if } C_t \le x \le 2C_t, \end{cases} \tag{7.9}$$

where C_t is the length of each sampling bucket. For the simple random sampling technique with the sampling rate r, H_s can be expressed as

$$H_s(i) = Pr(T_i = i) = (1-r)^{i-1}r. \tag{7.10}$$

Under the assumption that the process $X(t)$ is wide sense stationary, we have

$$\begin{aligned} R_g(\tau) &= E\left(g(t)g(t-\tau)\right) \\ &= E\left(X(t+t_0)X(t+t_0-u)\right) \\ &= E\left(X(t)X(t-u)\right) \\ &= E\left(E\left(X(t)X(t-u)|u\right)\right) \\ &= \sum_{u=0}^{\infty} R_X(u)p(u), \end{aligned} \tag{7.11}$$

where $u = \sum_{i=1}^{\tau} T_i$ and $p(u)$ is the probability mass function of u. Note that $p(u)$ is the τth order convolution of $H_s(u)$, which we denote as $k(u,\tau)$ (as it is a function of both u and τ). Now we are in a position to derive the sufficient and necessary condition.

Theorem 7: Given any wide sense stationary (WSS) process $X(t)$, the sampled process $g(t)$ obtained from a sampling technique with h_s or H_s has the same second order statistics as the original process asymptotically if and only if the following condition holds

$$\sum_{u=0}^{\infty} R_X(u)k(u,\tau) \sim R_X(\tau), \tag{7.12}$$

where $k(u,\tau)$ is the τth order convolution of $H_s(u)$. *Proof:* By Eq. 7.11 we know that $g(t)$ retains the same second order statistics of $X(t)$ asymptotically, if and only if $R_g(\tau) \sim R_X(\tau)$, as $\tau \to \infty$, and hence the conclusion.

Although Theorem 7 gives a sufficient and necessary condition for a sampling technique to retain the second order statistics of the original process, it cannot be readily applied, since $k(x, \tau)$ usually does not have a closed form, except for several extremely simple cases (e.g., for example the systematic sampling, in which $k(x, \tau) = \delta(x - \tau C_t)$, where $\delta()$ is the impulse *Dirac* function and C_t is the constant sampling interval).

In order to be able to apply Theorem 7, we propose a numerical method to calculate $k(x, \tau)$:

(S1) Calculate the Fourier transform of $H_s(x)$, $H_s(\omega)$. [2]

(S2) Let the Fourier transform of $k(x, \tau)$ (in terms of x) be $K(\omega, \tau)$. Then $K(\omega, \tau) = H_s(\omega)^\tau$.

(S3) Obtain $k(x, \tau)$ by deriving the inverse Fourier transform (IFT) of $K(\omega, \tau)$.

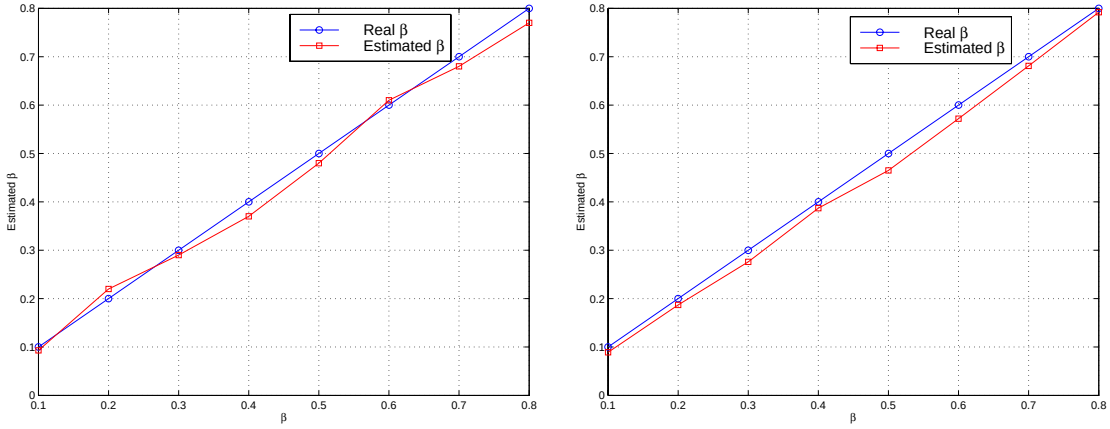
With $k(x, \tau)$, we can then calculate the left hand side of Eq. (7.12), and compare it against $R_X(\tau)$ as $\tau \to \infty$. Since fast algorithms exist for both the Fourier and inverse Fourier transform, the above method provides a fast and reliable test in evaluating Eq. (7.12).

To validate the above proposed method for applying Theorem 7, we apply it to check the random stratified and simple random sampling techniques, and give the results in Fig. 7.3. As shown in Fig. 7.3, the estimated and real value of β agree extremely well, which is consistent with the derivation in Sections 7.2.2–7.2.3.

7.3 The Average Variance of Sampling Methods

Due to the randomness nature of stratified random sampling and simple random sampling, sampling results vary from one sampling instance to another, even if multiple instances of sampling are taken simultaneously and the same sampling rate is applied in each instance. Here by "instance," we mean each experiment made to take samples for a specific time interval. Even for systematic sampling, different starting sampling points may lead to different sampling results. If the variance of sampling results obtained from multiple instances is large, then one cannot rely on a single sampling instance to infer the entire process. To evaluate different sampling techniques in this aspect, we use the average variance of sampling results $E(V)$ as the index. Recall that $E(V)$ is defined as follows in

[2] In the case that $H_s(x)$ cannot be expressed in a closed form, the proposed numerical method cannot be used to apply Theorem 7.



(a) stratified random sampling (b) simple random sampling

Figure 7.3: Estimated and real values of β under stratified random sampling and simple random sampling.

Section 1: let $\bar{X}$ be the real mean of the parameter of interest in the original process, and X_i be the sampled result in the ith instance of sampling (i.e., the ith experiment). Then the average variance is defined as $E(V) = E[E[(X_i - \bar{X})^2]]$.

Let V_{sy}, V_{rs} and V_{ran} denote, respectively, the variance of sampling results of systematic, stratified random and simple random sampling. To compare the three sampling techniques with respect to the average variance of sampling results, we leverage the results from [29] (Theorem 8.6):

Theorem 8: For a random process $X(t)$, with mean μ, variance σ^2, and autocorrelation function $R(\tau)$, if the following condition holds,

$$\delta_\tau = R(\tau + 1) + R(\tau - 1) - 2R(\tau) \geq 0, \tag{7.13}$$

we have $E(V_{sy}) \leq E(V_{rs}) \leq E(V_{ran})$.

The result in Theorem 8 is actually quite intuitive. For systematic sampling, as the sampling interval remains unchanged among different sampling instances, the same second order statistic structure (e.g., the autocorrelation function) is retained. For the other two sampling techniques, different sampling instances have different second order statistic structures, although in the long run, they follow the same decreasing rule.

Theorem 8 gives a sufficient condition (Eq. (7.13)) in evaluating the three sampling techniques

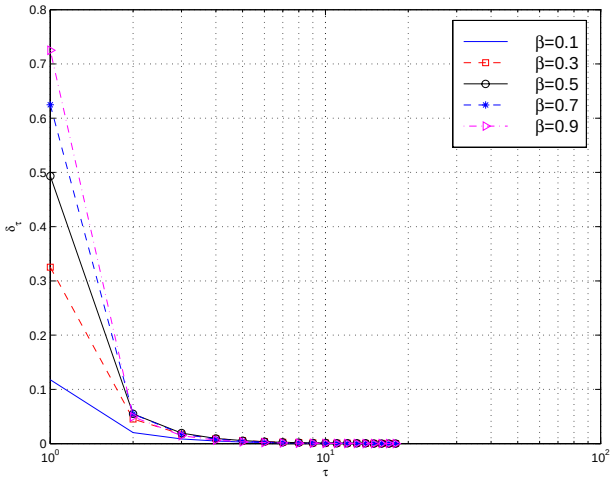
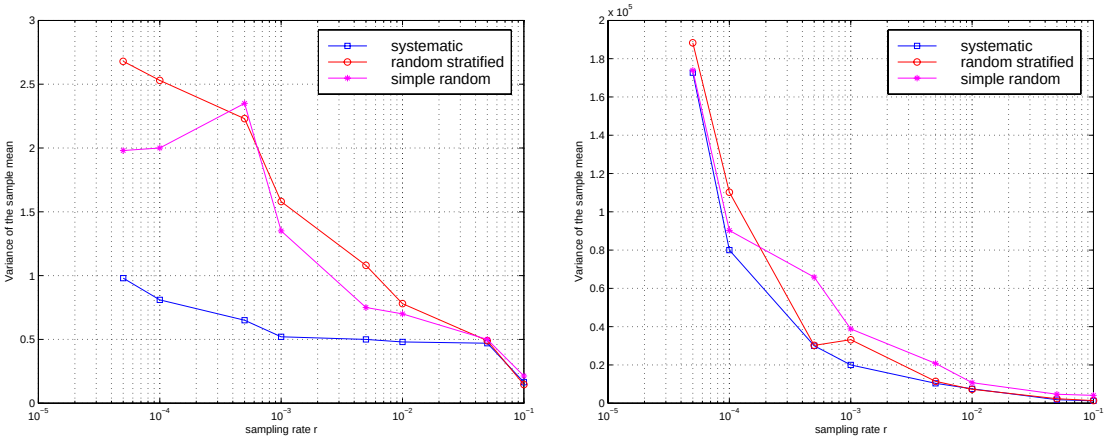


Figure 7.4: δ_τ versus τ for different values of β.

with respect to $E(V)$, given that the original process has finite mean and variance. To leverage Theorem 8, we first check whether the condition in Eq. (7.13) holds for a self-similar process. Using the fact that $R(\tau) \sim const \cdot \tau^{-\beta}$, we calculate δ_τ for different values of β and depict it in Fig. 7.4. As shown in Fig. 7.4, δ_τ is always positive regardless of the value of β, i.e., the condition in Eq. (7.13) holds.

In applying Theorem 8 we also need to verify if the process has finite mean and variance. A self-similar process (with $\alpha \in (1, 2)$) has finite mean, but its variance goes to infinity as time goes to infinity. However, in practice we often consider finite time periods, and hence we conjecture the above condition is still valid. To verify the conjecture, we carry out experiments and measure the average variance of sampling results (under the three techniques) on both synthetic and real Internet traffic. In this experiments, we generate in *ns-2* self-similar traffic with Hurst parameter equal to 0.80 using the on-off model, where the on/off periods have heavy-tailed distributions with shape parameter $\alpha = \beta+1$, $1 \leq \alpha \leq 2$. We also obtain real Internet traces from Lucent Technologies Bell Labs [88]. The set of traces was obtained on March 8, 2000, is in the *tcpdump* format, and contains detailed packet level information for hundreds of pairs of end hosts. The traces last for about 40 minutes and contains millions of packets. Fig. 7.5 shows the results. Note that Fig. 7.5 (b) gives the result for one of the trace sets with the Hurst parameter 0.62. Results for the other sets (that correspond to different servers) show similar trends and are not shown here. As shown



(a) result for synthetic data (b) result for real Internet traffic

Figure 7.5: The average variance of sampling results (under systematic, stratified random, and simple random sampling) on both synthetic and real Internet traffic.

in Fig. 7.5, systematic sampling does give the smallest average variance.

Although systematic sampling does capture the Hurst parameter and provide sampling results of small variance, we show in the next section that it provides very biased estimates of the real mean for a self-similar process. Due to this drawback, we then devise a new variation of systematic sampling to improve the accuracy of sampling results, while retaining all of its good properties. In the subsequent discussion, we will focus on systematic and simple random sampling, as stratified random sampling is a variation of systematic sampling.

7.4 Biased Systematic Sampling for Heavy-Tailed Traffic

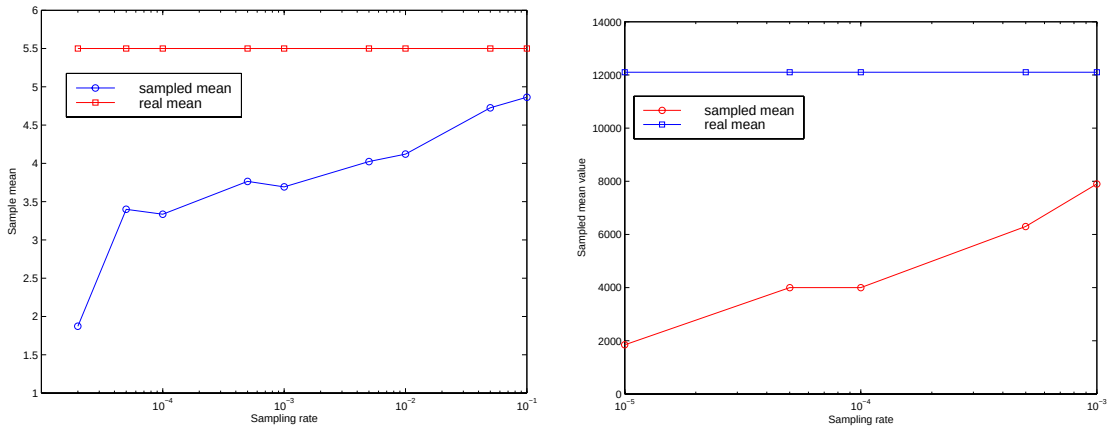
In this section, we first show that both systematic sampling and simple random sampling fail to provide a good estimate of the actual mean for a self-similar process (e.g., Internet traffic). Then based on an important observation on self-similar processes (validated through experiments), we propose a new extension of systematic sampling to remedy the above deficiency. The dilemma here is that the major portion of a self-similar process consists of "small values," while a small portion of "extremely large values" contributes to the majority of the volume of the entire process (which in turn dramatically affects the mean of the process). Due to the massive amount of Internet traffic and the storage limitation, the sampling rate and hence the number of samples cannot be too large,

but in order to capture the effect of these extremely large values (that occur not as often), one has to gather a large amount of samples. Similar observations have been made in the literature. For example, it has been reported in [34] that the steady-state behavior for self-similar workloads can be elusive, due to the fact that the average behavior depends on the presence of many small observations as well as a few large observations. The same observation has also been made in [100] on sampling Internet traffic, but no effective solution has been proposed to counter this problem.

7.4.1 Problem with Sampling a Self-Similar Process

By the central limit theorem (or the law of large numbers), the sampled mean can be used to approximate the real mean for any stationary process with finite mean and variance, as long as the sampling techniques are un-biased. It is well known that both simple random sampling and systematic sampling provide an un-biased estimator of the real mean for stationary processes with finite mean and variance as the number of samples goes to infinity. (In practice, a moderate number of samples suffice to provide a relatively good estimate of the real mean.) On the other hand, if the original process has infinite variance, e.g., a self-similar process, the law of large numbers does not hold, and the sampled mean approaches the real mean slowly, as the number of samples increases. As shown in [34], in order to achieve two-digit accuracy in the mean, the number of samples needed is up to 10^{22} for the case of $\alpha = 1.2$ (which corresponds to $H = 0.9$). Even for mild cases where $\alpha = 1.5$ ($H = 0.75$), still a million samples is required to achieve the desirable accuracy.

We carry out experiments to demonstrate the problem in the context of Internet traffic. In the experiments, we use the same set of synthetic and real Internet traffic traces used in Section 7.3. For synthetic trace, we change the sampling rate from 10^{-5} to 0.1, while for the real Internet trace, the sampling rate varies from 10^{-5} to 10^{-3}. (The reason why we used a smaller sampling rate is due to the large volume of Internet traces. In fact, a sampling rate of 10^{-3} is considered quite high, given the fact that tera-bytes of traffic is generated per day.) As shown in Fig. 7.6, in the case of synthetic traffic trace, the discrepancy between the real mean and the sampled mean (obtained even with a sampling rate of 0.1) is quite notable. The discrepancy becomes even more pronounced in the case of real Internet traces: the sampled mean obtained with a sampling rate of a 10^{-3} is approximately $\frac{2}{3}$ of the real mean, although in both cases the sampled mean increases steadily but



(a) result for synthetic data (b) result for real Internet traffic

Figure 7.6: The sampled mean and the real mean of a self-similar process versus different sampling rates.

slowly.

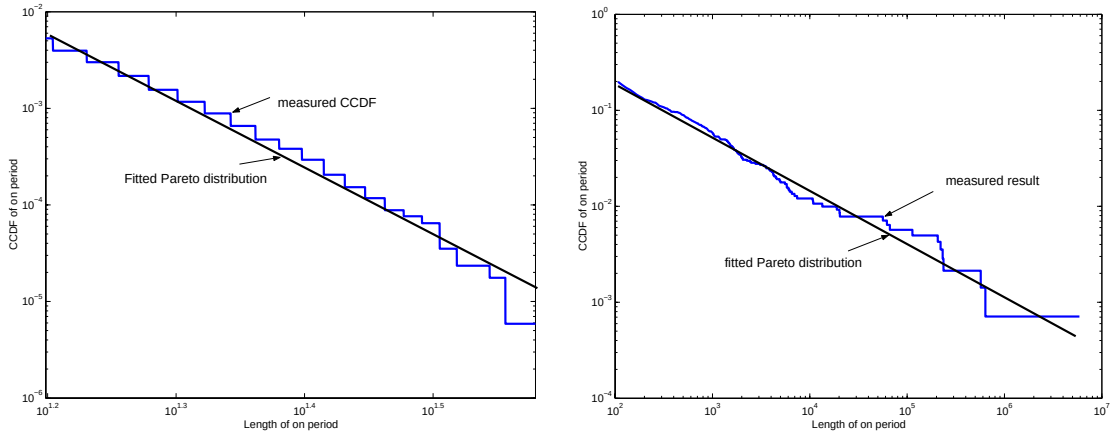
7.4.2 An Important Observation

As mentioned above, the reason why the sampled mean is always far less than the real mean for a self-similar process is that the major portion of a self-similar process consists of "small values," while a small portion of "extremely large values," albeit occurring less often, contributes to the majority of the volume of the entire process. Without use of a sufficiently high sampling rate, the large values are less likely to be sampled and hence the sampled mean is always less than the real mean. If one could instrument the sampling method to capture these extremely large values, the discrepancy between the sampled mean and the real mean can be reduced.

To instrument a sampling method to capture extremely large values, we need to identify where they occur. For a self-similar process $X(t)$, we define another on-off process $q(t)$ as:

$$q(t) = \begin{cases} 1, & \text{if } X(t) > a_{th}, \\ 0, & \text{otherwise,} \end{cases} \tag{7.14}$$

where a_{th} is a constant approximately of the same order of magnitude as the mean of $X(t)$, $\bar{X}$. The process $q(t)$ consists of bursts of 1s and 0s. The length of the 1-burst period is a random variable (which we denote as B).



(a) Synthetic trace (b) Real Internet trace

Figure 7.7: The CCDF of the 1-burst period B for the case of $\epsilon = 0.5$, where ϵ determines the onset value, α, of the 1-burst period ($a_{th} = \bar{X} \times \epsilon$).

We conjecture that due to the self-similar properties of $X(t)$, B is heavy tailed. Intuitively this conjecture is made based on the fact that a self-similar process contains concentrated periods of high activity and low activity, and hence once the process goes beyond a_{th}, the time interval B during which it continuously remains above a_{th} is heavy-tailed. To validate the conjecture, we again carry out experiments on both the synthetic and real Internet traces introduced in Section 7.3. In the experiments, we set $a_{th} = \bar{X} \times \epsilon$, where ϵ is called *normalized* a_{th} and varies from 0.5 to 1.5. For each fixed value of ϵ, we measure B and fit its CDF to the most widely used heavy tailed distribution, the Pareto distribution. Fig. 7.7 gives the results for $\epsilon = 1.0$. The fitted Pareto distribution has the shape parameter $\alpha = 1.3$ for the case of synthetic traces, while the shape parameter $\alpha = 1.65$ for the case of real Internet traffic traces. For different values of ϵ, the value of α changes mildly from 1.2 to 1.8, but the heavy-tailed nature of B remains unchanged.

7.4.3 Detailed Description and Analysis of Biased Systematic Sampling

In this section, we propose, based on the observation made in Section 7.4.2, a new variation of systematic sampling, called *biased systematic sampling* (BSS), that captures extremely large values more faithfully. Specifically, BSS is essentially systematic sampling with a sampling interval C_t, except that when a sample is greater than a threshold a_{th}, L_x extra samples are *evenly* taken in the current sampling interval C_t (i.e., the sampling interval for these extra samples is C_t/L_x). Among

these extra samples, we only keep those that are greater than a_{th} (which we henceforth call *qualified samples*).

Analysis The rational behind this design is as follows. A sample that is greater than a_{th} must fall in one of the 1-burst periods. Let the 1-burst period in which the sample falls be denoted as B. Suppose the sample is taken τ time units after the beginning of the 1-burst period B. We show given that B is heavily tailed, the probability that the next sample taken under BSS also exceeds a_{th} goes to 1 as τ goes to infinity. In other words, once a sample is taken with the value larger than a_{th}, it is highly possible that the values thereafter will still be larger than a_{th}. Specifically, such a probability can be expressed as

$$\begin{aligned} \wp(\tau) &= \Pr(q(\tau+1)=1 | q(t)=1, 1 \le t \le \tau) \\ &= 1 - \frac{\Pr(B=\tau)}{\Pr(B \ge \tau)}. \end{aligned} \tag{7.15}$$

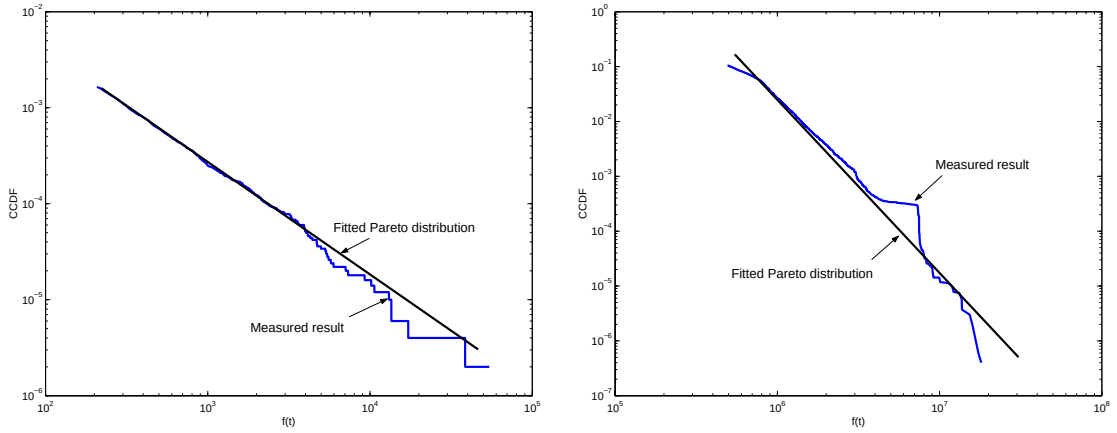
In the case that B is lightly tailed, e.g., the CCDF of B has an exponential tail, or $\Pr(B > x) \sim c_1 e^{-c_2 x}$, where c_1 and c_2 are two positive constants, Eq. 7.15 can be re-written as

$$\wp(\tau) \sim 1 - \frac{c_1 e^{-c_2 \tau} - c_1 e^{-c_2(\tau+1)}}{c_1 e^{-c_2 \tau}} = e^{-c_2}. \tag{7.16}$$

That is, in the case that B is lightly tailed, the probability that the samples taken exceed a_{th} does not become larger conditioning on the event that a sample has been identified to exceed a_{th}. In the case that B is heavily tailed, we have $\Pr(B > x) \sim c x^{-\alpha}$, where $1 \le \alpha \le 2$ is the index of heavy-tailedness of the process, and hence

$$\wp(\tau) \sim 1 - \frac{c\tau^{-\alpha} - c(\tau+1)^{-\alpha}}{c\tau^{-\alpha}} = (\frac{\tau}{\tau+1})^{\alpha}. \tag{7.17}$$

That is, $\wp(\tau) \to 1$, as $\tau \to \infty$. This implies given that B is heavily tailed, once a sample exceeds a_{th}, with a high probability the process will keep on large values. This lays the theoretical base for BSS, and ensures all the extra samples taken do increase the chance of capturing extremely large values.



(a) Synthetic trace with $\alpha = 1.5$ (b) Real Internet trace with $\alpha = 1.71$.

Figure 7.8: The CCDF of $X(t)$ and fitted Pareto distribution for synthetic and real Internet traces.

Parameters Setting in BSS There are two important parameters used in BSS: the on-set threshold a_{th} and the number, L_x, of extra samples in each sampling interval C_t. In what follows, we perform an analysis on the relationship of these two parameters. In the analysis we assume that $X(t)$ follows a Pareto distribution with shape parameter α. This assumption is reasonable which has been demonstrated in [21]. Furthermore, we use the synthetic and real Internet traces as have been used in Section 7.5.2 to show that $X(t)$ is heavy-tailed. The results are shown in Fig. 7.8. We show the CCDF of $X(t)$ and fit it to a Pareto distribution with shape parameter $\alpha = 1.5$ and $\alpha = 1.71$ for the synthetic and real traces respectively.

Let X_r, X_s, and X_{bss} denote, respectively, the real mean, the sampled mean under systematic sampling, and the sampled mean under BSS. By the property of the heavy tail distribution, $X_r = \frac{\ell\alpha}{\alpha-1}$, where ℓ is the lowest value the original process can take. Also, let the difference, η, between X_r and X_s be defined as

$$\eta = 1 - \frac{X_s}{X_r}. \tag{7.18}$$

Since the original process is self-similar, the sampled process is also self-similar with the same shape parameter α (Section 7.2). As a result, the probability that a sample is greater than a_{th} is $(\ell/a_{th})^\alpha$, where ℓ is the lowest value the original process can take. In other words, approximately $(\ell/a_{th})^\alpha \times N$ samples exceeds the on-set threshold, and trigger the operation of taking L_x extra samples. By a similar line of reasoning, approximately $(\ell/a_{th})^\alpha \times L_x$ samples (out of the L_x extra samples) exceed

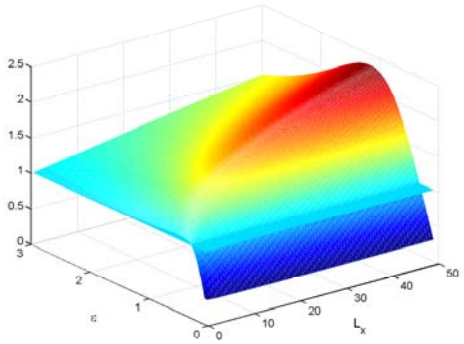


Figure 7.9: The relationship among L_x, ϵ and ξ in BSS.

the threshold a_{th}, and are classified as (*qualified* among all the extra samples taken). The sampled mean of the set of *qualified samples* taken is *approximately* $\frac{a_{th}\alpha}{\alpha-1}$.

Now the sampled mean, X_{bss}, under BSS can be expressed as

$$X_{bss} = \frac{N \cdot X_r + (\frac{\ell}{a_{th}})^{2\alpha} \cdot N \cdot \frac{a_{th}\alpha}{\alpha-1} \cdot L_x}{N + L_x \cdot (\frac{\ell}{a_{th}})^{2\alpha} \cdot N}. \tag{7.19}$$

Recall that $X_r = \frac{\ell\alpha}{\alpha-1}$ is the real mean, and hence Eq. (7.19) can be rewritten as

$$\begin{aligned} X_{bss} &= \frac{\ell\alpha}{\alpha-1} \cdot \frac{1 + L_x(\ell/a_{th})^{2\alpha-1}}{1 + L_x(\ell/a_{th})^{2\alpha}} \\ &\triangleq X_r \cdot \xi, \end{aligned}$$

where

$$\xi = \frac{1 + L_x(\ell/a_{th})^{2\alpha-1}}{1 + L_x(\ell/a_{th})^{2\alpha}} \tag{7.20}$$

is the *bias parameter*. If $\xi = 1$, then BSS is an unbiased sampling method. Given the values of ℓ and α, ξ is determined by L_x and a_{th}. In Fig. 7.9 we show the relationship between ξ L_x, and the normalized threshold $\epsilon = a_{th}/X_r$. The intersection of the plane of $\xi = 1$ and the surface of ξ gives the set of parameters that makes BSS unbiased. In particular, given any fixed value of L_x, there exists only one intersection point along the ϵ axis: $\epsilon = \frac{\alpha-1}{\alpha}$. This solution is, however, not feasible in practice, because $\epsilon = \frac{\alpha-1}{\alpha}$ for $1 \leq \alpha \leq 1.2$ is very small and suggests extra samples be taken in virtually every sampling interval. This translates to sampling at a very high rate.

A remedy to this problem is to allow BSS to be biased ($\xi > 1$). A key step along this direction

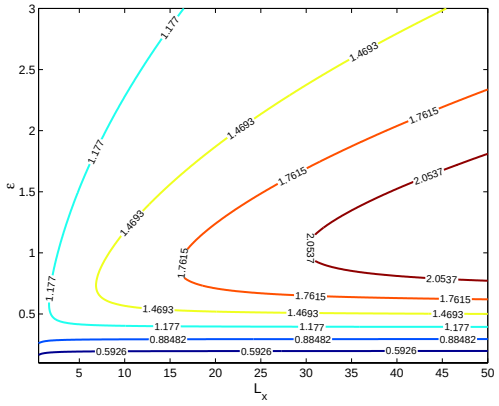


Figure 7.10: The contour of ξ.

is to determine the value of ξ. Note that $X_{bss} = X_r \cdot \xi$ (Eq. (7.20)) holds only when $N \to \infty$. In the case that N is finite, $X_{bss} \approx X_s \cdot \xi$. In order to have X_{bss} approach X_r in the finite-N case, we set $X_s \cdot \xi = X_r$ or $\xi = \frac{1}{1-\eta}$.

Tuning L_x and a_{th} in the case that η is known: If η is known, we can calculate $\xi = \frac{1}{1-\eta}$ and select appropriate values of L_x and a_{th} by intersecting the surface of ξ in Fig. 7.9 with the plane $\xi = \frac{1}{1-\eta}$. Fig. 7.10 gives the contour of ξ. The label on each contour curve indicates the value of ξ. Since all the points on the same contour curve render the same value of ξ, we can set one of the two parameters first and determine the other one accordingly.

Now to further determine the values of L_x and a_{th}, we take into account of the number of *qualified samples*, $N \cdot L_x \cdot (\frac{\ell}{a_{th}})^{2\alpha}$. This can be considered as the *overhead* in BSS. L_x and a_{th} should be so chosen that the number of qualified samples is as few as possible. That is, one should avoid the combination of a small value of ϵ and a large value of L_x. In our performance evaluation study, we set $\epsilon = 1$.

In summary, given the value of η, ξ can be calculated as $\frac{1}{1-\eta}$. Given the calculated ξ, the relation between a_{th} and L_x can be determined. Appropriate values of a_{th} and L_x can then be determined with the consideration of reducing the overhead of BSS as much as possible.

Tuning L_x and a_{th} without the knowledge of η In reality, as X_r is not known *a priori*, η cannot be readily obtained. In what follows, we discuss how to set the value of a_{th} given the value of ϵ, without the knowledge of X_r. Then we determine the value of L_x.

To determine the value of a_{th}, we propose an on-line tuning scheme. Before applying BSS, we first take N_{pre} samples (which we call *pre-samples*) from which we obtain an initial estimate of the mean and set the value of a_{th} accordingly. Then BSS commences, and we set the value of a_{th} as $a_{th} = E(X_{bss,i}) \times \epsilon$, where $X_{bss,i}$ is the sampled mean of the sample set that contains all the samples up to and including the ith *regular* sample (i.e., the set includes the *pre-samples*, the i samples and all the *qualified samples* taken so far). Note that during the course of taking extra qualified samples in a sampling interval, the value of a_{th} is not updated, since whether or not to take extra samples in a sampling interval should be based on the same threshold. Only by the end of a sampling interval when the next *regular* sample is to be taken will the value of a_{th} be updated as $E(Y_i) \times \epsilon$.

Given the value of a_{th}, the value of η is needed to set an appropriate value of L_x. In the lack of the η value, we estimate it from the sampling rate r as follows. As shown in [100] (Chapter 3), if we define

$$V_n = N^{1-1/\alpha}(X_s - X_r), \tag{7.21}$$

then

$$V_n \to \varphi_\alpha, \quad \text{in distribution}, \tag{7.22}$$

where φ_α is an α-*stable* distribution. That is, V_n converges in distribution for large values of N, i.e., $|X_s - X_r| \sim N^{1/\alpha-1}$. Hence,

$$\eta = \frac{|X_r - X_s|}{X_r} \sim \frac{N^{1/\alpha-1}}{X_r}. \tag{7.23}$$

Let N_{total} be the total number of points in the original processes, and r the systematic sampling rate. Then $N = N_{total} \cdot r$, and

$$\eta \sim C_s \cdot r^{1/\alpha-1}, \tag{7.24}$$

where $C_s = \frac{N_{total}^{1/\alpha-1}}{X_r}$ is a constant less than 1 for $1 \leq \alpha \leq 2$. In our experimental study, we find that for synthetic traces ($\alpha = 1.5$), $C_s \in (0.08, 0.15)$ while for real traces ($\alpha = 1.66$), $C_s \in (0.05, 0.1)$.

In summary, Eq. (7.24) is used to estimate the value of η. With the value of η, one can obtain $\xi = \frac{1}{1-\eta}$. By plugging in both the values of ξ and a_{th} in Eq. (7.20), one can obtain the value of L_x.

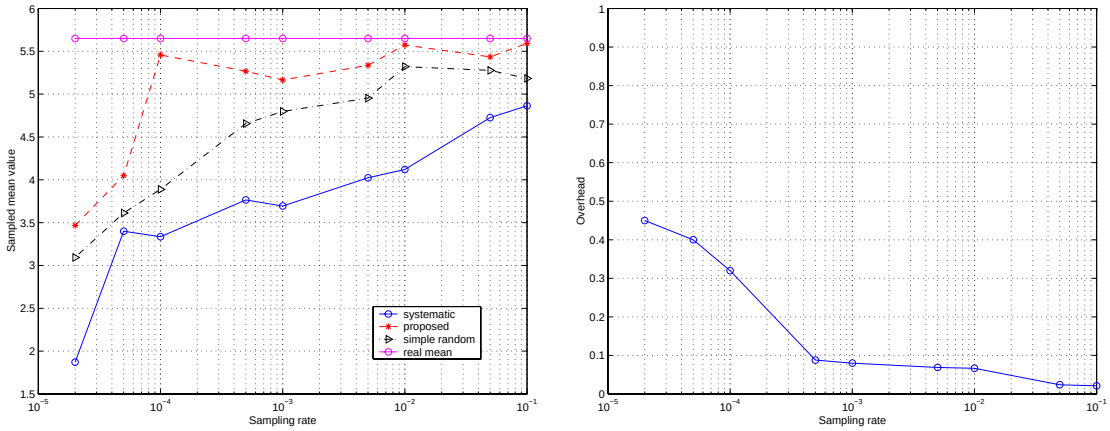
7.5 Performance Evaluation

To evaluate the performance of BSS, we have carried out several sets of experiments on both synthetic and real Internet traces. As the increase in the accuracy of the sampled mean in BSS is obtained at the cost of sampling more "biased" samples of larger values, we use the following three metrics to evaluate BSS: (1) the sampled mean (accuracy); (2) the sampling *overhead*, defined as the ratio of the number of *qualified samples* to the number of samples taken by systematic sampling; and (3) the efficiency e, defined as $e = \frac{1-\eta}{log(N_t)}$ and N_t is the total number of samples (including both the samples normally taken in systematic sampling and the *qualified samples* taken in BSS). In addition to the above three metrics, we also verify whether the sampled process has the same Hurst parameter as the original process and calculate its average variance. The performance evaluation is made by comparing BSS against systematic and simple random sampling. As stratified random sampling is a variation of systematic sampling and yields similar performance as the latter, we do not include it in the comparison study.

7.5.1 Performance w.r.t. Sampled Mean, Overhead and Efficiency

We use the same traces as in Section 7.3. For synthetic traces, we set the shape parameter of the on/off periods to be $\alpha \in (1.2, 1.6)$. Figures 7.11–7.12 give the sampled mean obtained by systematic sampling, simple random, and BSS ((a)), and the sampling overhead incurred in BSS ((b)) for both the synthetic traces and real Internet traces. Note that the result shown in Fig. 7.11 is for the synthetic trace with $\alpha = 1.3$ and mean value 5.68 kbytes/second, while that in Fig. 7.12 is for the Internet trace with the real mean rate 1.21×10^4 bytes/second and the (measured) Hurst parameter 0.62. (Results for the other traces exhibit similar trends and hence are not shown here.) As shown in Fig. 7.11 (a), BSS generates much more accurate sampled means than the other two sampling techniques. The performance improvement is especially pronounced when the sampling rate is as small as 10^{-4}. As shown in Fig. 7.11 (b), the overhead is below 0.2 for larger sampling rates ($\geq 10^{-4}$) and below 0.5 for smaller sampling rates, while $1 - \eta$ (Section 7.4.3) is 0.922 for BSS and 0.66 and 0.81 for systematic sampling and simple random sampling, respectively. Similar conclusions can be made in Fig. 7.12, except that the sampling overhead is around 0.2.

Fig. 7.13 compares BSS against systematic sampling and simple random sampling with respect



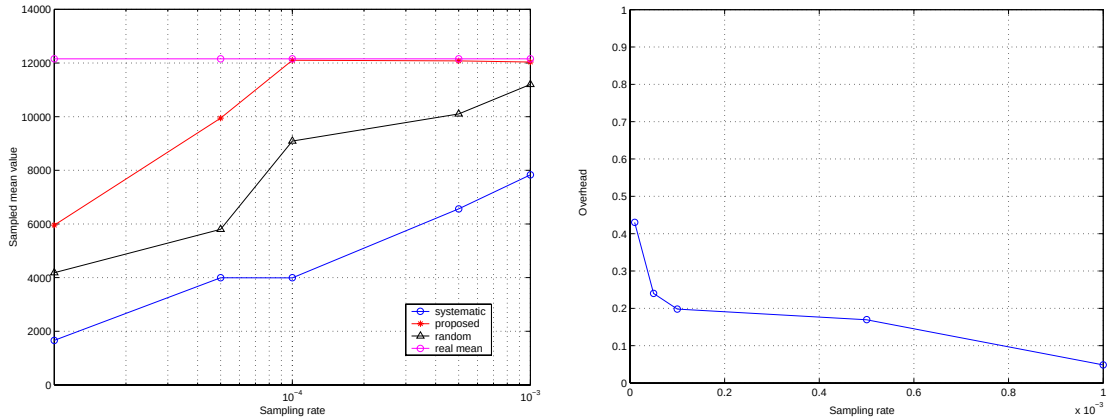
(a) Sampled mean (b) Sampling overhead

Figure 7.11: The sampled mean obtained by systematic sampling, simple random, and BSS ((a)), and and the sampling overhead incurred in BSS ((b)) for synthetic traces.

to the efficiency e for synthetic traces. BSS achieves higher efficiency than the other two sampling techniques. The average e for BSS is 0.36, while that for systematic and simple random sampling is 0.26 and 0.3, respectively, i.e., BSS achieves a performance gain of 40% and 20%, respectively, as compared to systematic and simple random sampling.

7.5.2 Performance w.r.t. Hurst Parameter and Average Variance

To verify whether or not BSS captures the Hurst parameter accurately, we use synthetic traffic with β ($H = \frac{2-\beta}{2}$) varying from 0.1 to 0.8 and give the result in Fig. 7.14. The Hurst parameter of the synthetic traces is calculated using a wavelet based tool provided by Abry *et al.* [111]. As shown in Fig. 7.14, the sampled process has the same value of β (and hence the same Hurst parameter) as the original process. This is not surprising, as BSS is a variation of static systematic sampling and the extra samples taken in each sampling interval are also taken in a systematic sampling fashion in each interval C_t. As a result, the sampled process generated by BSS keeps the same autocorrelation function as that generated by systematic sampling (which in turns is the same as that of the original process, Section 7.2.1). Finally, Fig. 7.15 gives the the average variances of BSS and systematic sampling for both synthetic and real Internet traces. As shown in the figure, the average variances of these two methods almost overlap completely. This is not surprising due to the same reason stated above.



(a) Sampled mean (b) Sampling overhead

Figure 7.12: The sampled mean obtained by systematic sampling, simple random, and BSS ((a)), and and the sampling overhead incurred in BSS ((b)) for real Internet traces.

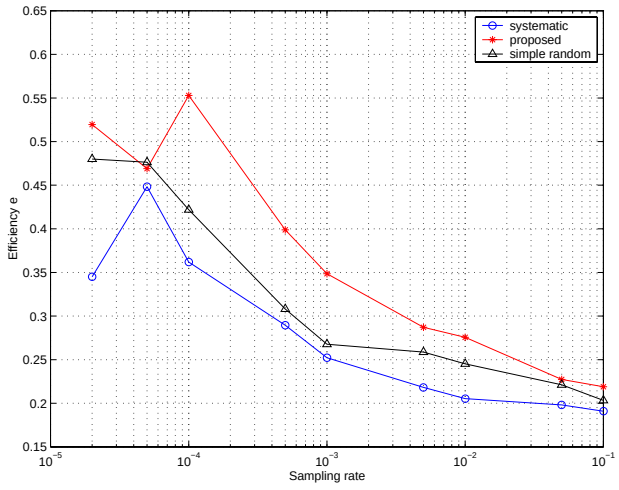


Figure 7.13: The efficiency of systematic sampling, simple random, and BSS for synthetic traffic.

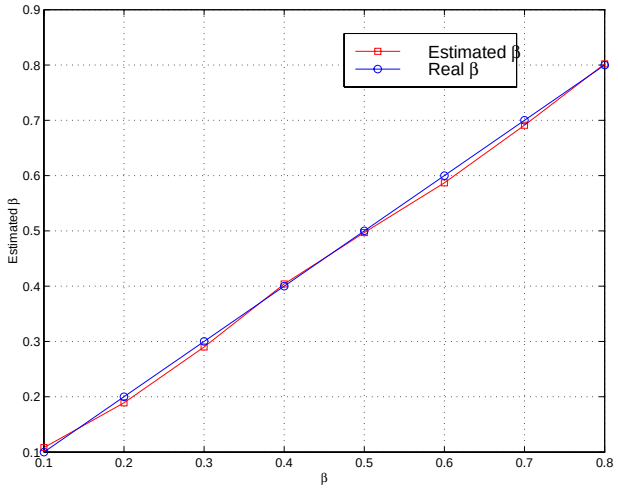
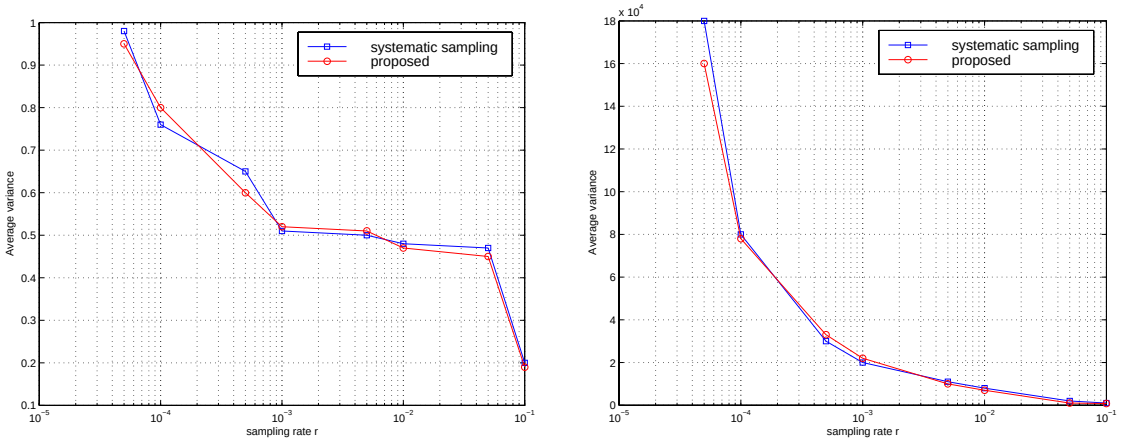


Figure 7.14: The β values of the sampled process generated by BSS and and the real process.



(a) Average variance for synthetic traces (b) Average variance for real Internet traces

Figure 7.15: The average variances of BSS and systematic sampling.

7.6 Summary

In this chapter, we have investigated several important issues in employing sampling techniques for measuring Internet traffic. We show that while all three sampling techniques can accurately capture the Hurst parameter (second order statistics) of Internet traffic, they fail to capture the mean (first order statistics) faithfully, due to the bursty nature of Internet traffic. We also show that static systematic sampling renders the smallest variation of sampling results in different instances of sampling (i.e., it gives sampling results of high fidelity). Based on an important observation, we then devise a new variation of systematic sampling, called *biased systematic sampling (BSS)*, that gives much more accurate estimates of the mean, while keeping the sampling overhead low. Both the analysis on the three sampling techniques and the evaluation of BSS are performed on both synthetic and real Internet traffic traces. The performance evaluation shows that BSS gives a performance improvement of 40% and 20% (in terms of efficiency) as compared to static systematic and simple random sampling.

An important lesson learned from the work is that although un-biased sampling methods are usually preferred for processes with finite means and variances (where the law of large numbers guarantees that the sampled mean approaches the real mean exponentially fast as the number of samples increases), it may not be the case for a process with an infinite variance (e.g., self-similar Internet traffic with the Hurst parameter larger than 0.5). Due to the heavy-tailedness inherited in the self-similar process, the speed for the sampled mean to converge to the real mean is extremely slow, and therefore un-based sampling techniques often render un-satisfactory results. In this case, a biased sampling method is actually desirable. By biasing toward the *large* values of the process, one can reduce the discrepancy between the sampled mean and the real mean. In this chapter we make a case where a biased sampling method outperforms un-biased ones.

Chapter 8

Conclusion and Future Work

In this thesis we study the scaling behavior of Internet traffic by tracing the source of the scaling behavior and proposing a hierarchical model for the sake of characterizing the Internet traffic. Then, we exploit the scaling behavior (self-similar/LRD property) of Internet traffic for resource and traffic management. Our major contributions are:

T1. The scaling property (self-similarity, LRD, multifractality) of the Internet traffic comes from the infrastructure of the protocol hierarchy of IP networks, where the Internet traffic is generated and modulated. By proposing a hierarchical model, we make it clearer how these complicated phenomena are generated, and to what extent they affect the queuing behavior of the Internet traffic.

T2. The nontrivial autocorrelation structure of the Internet traffic can be readily exploited to do accurate prediction. The prediction results can be applied in active queue management, so that the packet drop probability depends not only on the queue length but also on the future incoming traffic obtained by prediction. They can also be used to tune the TCP's AIMD phases so as to improve the attainable throughput and reduce packet loss.

T3. The scaling property, especially the scale-invariant autocorrelation structure of the Internet traffic shows its power in the domain of network end-to-end measurements. Specifically, the probe packet pattern can be tuned to bring us accurate estimation of the cross traffic on an end-to-end basis while the cost (the number of probe packets) is mild.

T4. In sampling a self-similar/LRD process with infinite variance, traditional sampling methods cannot provide satisfactory sampling mean value. The thesis shows that the inherent heavy-

tailedness possessed by the LRD traffic helps us to implement a biased systematic sampling (BSS) which amortizes the drawback of traditional sampling methods and give accurate estimate of the important statistics (first and second order moments).

8.1 Future Work

Our future work lies in the following aspects:

Refine exiting work In this thesis, to make the problem tractable we make several assumptions. Such as, in Chapter 3 we assume all the ON and OFF periods follow Pareto distributions (although we validate them using Internet traces, we cannot exhaust all the cases); in Chapter 6, we assume that there exists only one bottleneck link on the end-to-end path and the probe packets will not be queued before or after the bottleneck link; in Chapter 7, we propose a conjecture (the heavy-tialedness of the 1-burst period) without proving it. Study of the cases when the assumptions do not hold will be the first aspect of our future work.

Infinite variance process Throughout the thesis, we assume the Internet traffic has finite mean and variance. Recent measurements and study [58] show that $\alpha - stable$ network traffic model with infinite variance would be better than any other self-similar models in simulations. These $\alpha - stable$ ($S\alpha S$) processes make traditional covariance based tools, such as, the *LMMSE*-based methods nonsense. The research regarding $\alpha - stable$ model forms another part of our future work which consists of three aspects: first, the validation of the model, i.e., to what extent and under what circumstances the real Internet traffic can be faithfully modeled as a $\alpha - stable$ process; second, how to estimated the 4 parameters in the models; third, how can we make usage of this model to facilitate resource and traffic control.

Will the scaling property last forever? After more than 10 years' research on the scaling property of the Internet traffic, one largely acknowledged faith is: the scaling property comes from the Internet infrastructure itself. In this thesis we also make efforts to study this issue and a hierarchical model is proposed. One related open issue is: will this scaling property change with the development of Internet techniques? Especially, we have seen the rapid development of wireless networks (WLAN) and the future Internet will be a hybrid of both wireline and wireless networks.

Then, a possible future research direction is to verify whether or not the scaling property still holds in the network traffic. Exploitation on this direction can tell us more insights into the properties of the Internet and the traffic it generates.

Appendix A

Fractional Model-Based Predictors

A.1 Fractional Brownian Motion Model:

A normalized FBM with the Hurst parameter $H \in [1/2, 1)$ is a time series Z_t, $t \in (-\infty, \infty)$, characterized by the following properties: (i) Z_t has stationary increments; (ii) $Z_0 = 0$, and $E[Z_t] = 0$ for all t; (iii) $E[Z_t^2] = |t|^{2H}$ for all t; (iv) Z_t has continuous paths; and (v) Z_t is Gaussian, i.e., all its finite-dimensional marginal distributions are Gaussian.

When $H = \frac{1}{2}$, FBM reduces to the standard Brownian motion. When $1/2 < H < 1$, FBM models time series with the LRD characteristic. It is easy to show that the covariance of increments in two non-overlapping intervals, $[t_1, t_2]$ and $[t_3, t_4]$ $(t_1 < t_2 \leq t_3 < t_4)$ is always positive and can be expressed as:

$$cov(Z_{t_2} - Z_{t_1}, Z_{t_4} - Z_{t_3}) = \frac{1}{2}((t_4 - t_1)^{2H} - (t_3 - t_1)^{2H} + (t_3 - t_2)^{2H} - (t_4 - t_2)^{2H})^M.$$

By virtue of the properties of FBM, the prediction of $Z_a (a > 0)$ based on $\{Z_t \mid t \in (-T, 0)\}$ is equivalent to the prediction of the difference $Z_{t+a} - Z_t$ (for any t) based on the differences $Z_t - Z_s, s \in (t - T, t)$ [96]. Hence, as shown in [96], the predictor, $\hat{Z}_{a,T} \stackrel{\triangle}{=} E(Z_a | Z_s, s \in (-T, 0))$, can be expressed as:

$$\hat{Z}_{a,T} = \int_{-T}^{0} g_T(a, t) dZ_t. \tag{A.1}$$

Note that $Z_{a,T}$ is estimated in Eq. (A.1) as a weighted sum of $\{Z_t \mid t \in (-T, 0)\}$ with weight

coefficients $g_T(a,t)$. For $T < \infty$, $t \in (0,T)$, $g_T(a,t)$ can be expressed as

$$g_T(a,t) = \frac{sin(\pi(H-\frac{1}{2}))}{\pi}(-t(T+t))^{-H+\frac{1}{2}} \int_0^a \frac{(\tau(\tau+T))^{H-\frac{1}{2}}}{\tau - t} d\tau.$$

The convergence of the above integration has been shown in [96] to be in the L^2-norm with respect to the Lebesgue measure on the real numbers. However, to on-line calculate $g_T(a,t)$, one has to on-line estimate the parameter H. It is also shown in [96] that

$$Var(E[Z_a|Z_s, s \in (-T,0)]) = Var(Z_a) \cdot H \cdot \int_0^{T/a} g_{T/a}(1,-s)\ ((1+s)^{2H-1} - s^{2H-1}) ds$$

and for $T = \infty$,

$$Var(E[Z_a|Z_s, s \leq 0]) = Var(Z_a)(1 - \frac{sin(\pi(H-1/2))\Gamma(3/2-H)^2}{\pi(H-1/2)\Gamma(2-2H)}),$$

where $\Gamma()$ is the gamma function. Finally, the relative variance of error is:

$$Var(Z_a - \hat{Z}_{a,\infty})/Var(Z_a) = \frac{sin(\pi(H-1/2))\Gamma(3/2-H)^2}{\pi(H-1/2)\Gamma(2-2H)}. \quad (A.2)$$

A.2 Fractional ARIMA Model:

The fractional ARIMA model is an extension of classic ARIMA models introduced by Box and Jenkins [16]. Because of their simplicity and flexibility, the ARIMA models have been widely used in time series analysis. Specifically, let the back-shift operator $\tilde{B}$ be defined as $\tilde{B}^n X_t \stackrel{\triangle}{=} X_{t-n}$, and p and q be two integer-coefficient polynomials $\phi(x)$ and $\psi(x)$ defined as:

$$\phi(x) = 1 - \sum_{j=1}^{p} \phi_j x^j, \quad \text{and} \quad \psi(x) = 1 + \sum_{j=1}^{q} \psi_j x^j,$$

where ϕ_j's and ψ_j's are coefficients such that all the roots of $\phi(x)$ and $\psi(x)$ are outside the unit cycle to ensure the stationarity of the time series. Moreover, let ϵ_t $(t = 1, 2, ...)$ be i.i.d normal random variables with zero mean and variance σ^2. Then, an $ARMA(p, q)$ model is defined as the stationary solution of

$$\phi(\tilde{B})X_t = \psi(\tilde{B})\epsilon_t, \tag{A.3}$$

i.e., X_t is expressed as a weighted sum of recent signal samples X_{t-1}, X_{t-2}, ..., X_{t-p}, and recent noise samples ϵ_t, ϵ_{t-1}, ..., ϵ_{t-q}, and the $ARIMA(p, d, q)$ model is defined as

$$\phi(\tilde{B})(1 - \tilde{B})^d X_t = \psi(\tilde{B})\epsilon_t, \quad d \geq 0.$$

X_t is called a fractional $ARIMA(p, d, q)$ time series if $-\frac{1}{2} < d < \frac{1}{2}$ and the relation of d and H can be expressed as $d = H - \frac{1}{2}$. Long-range dependency occurs when $\frac{1}{2} < H < 1$ or $0 < d < \frac{1}{2}$.

In the family of $ARIMA(p, d, q)$ models, $ARIMA(0, d, 0)$ is well studied due to its simplicity and tractability. Specifically, let X_t be a fractional $ARIMA(0, d, 0)$ time series with $-\frac{1}{2} < d < \frac{1}{2}$, then as reported in [68], the estimator of X_t (in the *LMSE* sense) is

$$\hat{X}_t = \sum_{j=1}^{k} \beta_{kj} \ X_{t-j}, \tag{A.4}$$

where the coefficients β_{kj} can be explicitly written as

$$\beta_{kj} = -\begin{pmatrix} k \\ j \end{pmatrix} \frac{\Gamma(j - d)\Gamma(k - d - j + 1)}{\Gamma(-d)\Gamma(k - d + 1)}, \tag{A.5}$$

and the mean square error of this predictor is

$$\sigma^2 \ = \ E\left(|X_t - \hat{X}_t|^2|\right) = \sigma_x^2 - \sum_{j=1}^{k} \beta_{kj} r(j), \tag{A.6}$$

where σ_x^2 is the variance of X_t, and $r(j)$ is the covariance function of X_t under the $FARIMA(0, d, 0)$

models and can be expressed as

$$r(j) = \sigma_x^2 \frac{(-1)^j \Gamma(1-2d)}{\Gamma(j-d+1)\Gamma(1-j-d)}.$$

Finally, the relative mean square error can be expressed as

$$\eta = \frac{\sigma^2}{\sigma_x^2} = 1 + \sum_{j=1}^{k} \left(\begin{array}{c} k \\ j \end{array} \right) \frac{\Gamma(j-d)\Gamma(k-d-j+1)(-1)^j \Gamma(1-2d)}{\Gamma(-d)\Gamma(k-d+1)\Gamma(j-d+1)\Gamma(1-j-d)} \tag{A.7}$$

Appendix B

Proof of Theorem 6

We first consider the case of two TCP connections sharing a bottleneck link. Let $r_i(k)$ denote the sending rate of connection i in the kth RTT, $k \geq 0$, η_i denote the ratio of $r_i(n+1)/r_i(n)$. We also write (for ease of notiation) the predicted throughput, $\hat{f}^{(m)}(n+1)$, attainable by connection i in the next interval as f_i. Under the assumption of accurate prediction, the optimal operational point is $(f_1 = C/2, f_2 = C/2)$ (Fig. B.1). Without loss of generality, suppose that the initial operational point is the cross point, $(r_1(n), r_2(n))$, with $r_1(n) > \frac{C}{2}$, $r_2(n) < \frac{C}{2}$, and

$$r_1(n) + r_2(n) \leq C$$

(the other cases can be similarly argued). According to the operations of PTCP, connection i will set its sending rate to $r_i(n+1) = C/2$. Hence, connection 1 multiplicatively increases its sending rate by a factor of $\eta_1 < 1$ and connection 2 increases its sending rate by a fact of $\eta_2 > 1$, and the operational point is dragged to the point $(C/2, C/2)$.

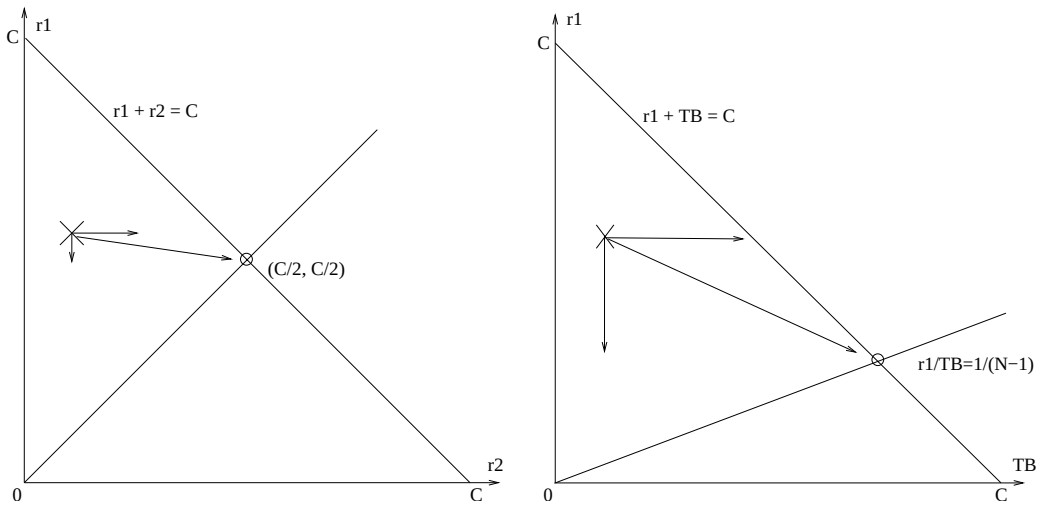
In the case of $N > 2$ connections sharing the bottleneck link, we have

$$\sum_{i=1}^{N} r_i(n) \leq C, \tag{B.1}$$

and

$$\sum_{i=1}^{N} r_i(n+1) = \sum_{i=1}^{N} \eta_i r_i(n) = C. \tag{B.2}$$

Without loss of generality, assume $\eta_1 \leq 1$ (the other cases can be similarly argued). Let $TB(n) =$



(a) $N = 2$ (b) $N > 2$

Figure B.1: How TCP-TP operates in the case of N connections sharing a bottleneck link.

$\sum_{i=2}^{N} r_i(n)$ represent the sending rate of the composite background traffic, and

$$I \triangleq \frac{\sum_{i=2}^{N} \eta_i r_i(n)}{\sum_{i=2}^{N} r_i(n)}. \tag{B.3}$$

Then Eqs. (B.1)–(B.2) can be rewritten as

$$r_1(n) + TB(n) \leq C, \tag{B.4}$$

and

$$\eta_1 r_1(n) + I \cdot TB(n) = C. \tag{B.5}$$

With $\eta_1 \leq 1$, we must have $I \geq 1$, and the multi-connection case reduces to the two-connection case with the fairness line $\frac{r_1(n)}{TB(n)} = \frac{1}{N-1}$, and the rest of the proof follows the two-connection case.

References

[1] A. M. Adas. Using adaptive linear prediction to support real-time VBR video under RCBR network service model. *IEEE/ACM Transactions on Networking*, Vol. 6, No. 5, October 1998.

[2] A. Adas and A. Mukherjee. On resource management and QoS guarantees for long range dependent traffic. In *Proc. of IEEE INFOCOM '95*, pages 779-787, 1995.

[3] R. Addie, M. Zukerman, and T. Neame. Fractal traffic: measurements, modeling and performance evaluation. In *Proc. of IEEE INFOCOM '95*, pages 977-984, 1995.

[4] S. Alouf, P. Nain, and D. Towsley. Inferring network characteristics via moment-based estimators. In *Proc. of IEEE INFOCOM '01*, pages 1045-1054, April 2001.

[5] V. Anantharam. Queueing analysis with traffic models based on deterministic dynamical systems. In *Proc. of 25th Allerton Conference on Communication, Control and Computing*, pages 233-241, 1996.

[6] F. M. Anjum, and L. Tassiulas. Fair bandwidth sharing among adaptive and non-adaptive flows in the Internet. In *Proc. of IEEE INFOCOM '99*, New York, USA, March 1999.

[7] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin. REM: active queue management. *IEEE Network Magazine*, Vol. 15, No. 3, pages 48-53, May/June 2001.

[8] P. Barford, and M. E. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proc. of Performance '98 ACM SIGMETRICS*, Madison WI, 1998.

[9] C. S. Barrus and T. W. Parks. DFT/FFT and convolution algorithms. *New York: John Wiley and Sons*, 1985.

[10] V. Benes. General stochastic processes in the theory of queues. Reading, MA:Addison Wesley, 1963.

[11] J. Beran. Statistics for long-memory process. Ch.4, CHAPMAN and HALL, 1994.

[12] N. H. Bingham, C. M. Goldie, J. L. Teugels. Regular variation. *Encyclopedia of Mathematics and its Applications*, Cambridge University Press, UK, 1987.

[13] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, July 1970.

[14] J. C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proc. of ACM SIGCOMM '93*, pages 289-298, 1993.

[15] D. P. Bovet and M. Cesati. Understanding the Linux Kernel. *O'Reilly & Associates, Inc.*,2001.

[16] G. E. P. Box and G. M. Jenkins. Time series analysis: forecasting and control. Holden Day, San Francisco, 1970.

[17] B. Braden *et al.* Recommendations on queue management and congestion avoidance in the Internet. RFC-2309, IETF, April 1998.

[18] F. Brichet, A. Simonian, L. Massoulie, D. Veitch. Heavy load queueing analysis with LRD on/off sources. *Self-Similar Network Traffic and Performance Evaluation*, Vol. 5, pages 115-141, 2000.

[19] R. Cáceres, N. G. Duffield, S. B. Moon and D. Towsley. Inferring link-level performance from end-to-end multicast measurements. In *Proc. of IEEE/ISOC Global Internet '99*, December 1999.

[20] R. Cáceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Journal on Selected Areas in Communications*, Vol. 45, No. 7, November 1999.

[21] J. Cao, W. S. Cleveland, D. Lin and D. X. Sun. The effect of statistical multiplexing on Internet packet traffic: theory and empirical study. *Bell Labs Tech. Report*, 2001.

[22] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. In *Proc. of ACM PERFORMANCE '96*, October 1996.

[23] C. Cetinkaya and E. Knightly. Egress admission control. In *Proc. of IEEE INFOCOM '00*, March 2000.

[24] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, June 1989.

[25] B. Y. Choi, J. Park and Z. L. Zhang. Adaptive random sampling for load change detection. *ACM SIGMETRICS*, 2002, (Extended Abstract).

[26] K. C. Claffy, G. C. Polyzos and H. W. Braun. Application of sampling methodologies to network traffic characterization. In *Proc. of ACM SIGCOMM '93*, September 1993.

[27] K. Claffy, G. Miller and K. Thompson. The Nature of the Beast: recent traffic measurements from an Internet backbone. In *Proc. of INET '98*.

[28] Cisco netflow. *http://www.cisco.com/warp/public/732/Tech/netflow*.

[29] W. G. Cochran. Sampling techniques. John Wiley & Sons, Inc., 1977.

[30] T. H. Cormen, C. E. Leiserson and R.L. Rivest. Introduction to algorithms. Ch.31, McGraw-Hill, 1994.

[31] D. R. Cox. Renewal theory. Imperial College University of London, 1967.

[32] I. Cozzani and S. Giordano. A Measurement based QoS evaluation. *IEEE SICON '98*, June 1998.

[33] M. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking*, Vol. 5, pages 835-846, December 1997.

[34] M. E. Crovella and L. Lipsky. Long-lasting transient conditions in simulations with heavy-tailed workloads. In *Proc. of the 1997 Winter Simulation Conference*.

[35] J. D. Deuschel and D. W. Strook. Large deviations. Academic Press, New York, 1994.

[36] N. G. Duffield and N. O'Connell. Large deviations and overflow probabilities for the general single server queue, with application. *Math. Proc. Cambridge Philos. Soc.*, Vol. 118, pages 363-374, 1995.

[37] N. G. Duffield, J. Horowitz, D. Towsley and T. Bu. Multicast-based inference of network-internal characteristics: accuracy of packet loss estimation. In *Proc. of IEEE INFOCOM '99*, April 1999.

[38] N. G. Duffield and F. Lo Presti. Multicast inference of packet delay variance at interior network links. In *Proc. of IEEE INFOCOM '00*, April 2000.

[39] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *Proc. of ACM SIGCOMM '00*, pages 271-282, August 2000.

[40] N. Duffield, C. Lund and M. Thorup. Charging from sampled network usage. In *ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[41] N. G. Duffield, C. Lund and M. Thorup. Properties and prediction of flow statistics from sampled packet streams. *ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[42] N. G. Duffield, C. Lund and M. Thorup. Estimating flow distributions from sampled flow statistics. In *Proc. of ACM SIGCOMM '03*, Germany, August 2003.

[43] R. Ellis. Large deviations for a general class of random vectors. Ann. Prob. Vol. 12, pages 1-12, 1984.

[44] A. Erramilli, R. Singh, and P. Pruthi. An application of deterministic chaotic maps to model packet traffic. *Queueing Syst.*, Vol. 20, pages 171-206, 1995.

[45] A. Erramilli, O. Narayan, and W. Willinger. Experimental queuing analysis with long-range dependent traffic. *IEEE/ACM Transactions on Networking*, April 1996.

[46] C. Estan and G. Varghese. New directions in traffic measurement and accounting. *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[47] A. Feldmann, A. C. Gilbert, W. Willinger. Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic. *Computer Communications Review*, Vol. 28, No. 4, pages 42-58, 1998.

[48] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford and F. True. Deriving traffic demands for operational ip networks: Methodology and experience. In *Proc. of ACM SIGCOMM '00*, pages 257-270, August 2000.

[49] Felix. Independent monitoring for network survivability. For more information, refer to http://www.bellcore.com/pub/mwg/felix/index.html.

[50] W. Feng, D. Kandlur, and K. Shin. Stochastic Fair Blue: a queue management algorithm for enforcing fairness. In *Proc. of IEEE INFOCOM '01*, April 2001.

[51] W. Feng, K. Shin, D. Kandlur, and D. Saha. A self-configuring RED gateway. In *Proc. of IEEE INFOCOM '99*, New York, USA, March 1999.

[52] D. R. Figueiredo, B. Liu, V. Misra, D. Towsley. On the autocorrelation structure of TCP traffic. Technical Report TR00-55, University of Massachusetts, Computer Science Department, Amherst, MA, 2000.

[53] S. Floyd and V. Jacobson. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, August 1993.

[54] S. Floyd. RED: discussions of setting parameters. http://www.aciri.org/floyd/REDparameters.txt, November 1997.

[55] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation based congestion control for unicast applications. In *Proc. of ACM SIGCOMM '00*, August 2000.

[56] J. Gao and I. Rubin. Multiplicative multifractal modeling of long-range dependent network traffic. *International Journal of Communication Systems*, Vol. 4, pages 783-801, 2001.

[57] Y. Gao, G. He, and J. C. Hou. On exploiting traffic predictability in active queue management. In *Proc. of IEEE INFOCOM '02*, June 2002.

[58] X. Ge, G. Zhu and Y. Zhu. On the testing for alpha-stable distributions of network traffic. *Computer Communications*, Vol. 27, No. 5, Pages 447-457, March 2004.

[59] L. Guo, M. Crovella, I. Matta. TCP congestion control and heavy tails. Technical Report BU-CS-2000-017, Boston University, Computer Science Department, Boston, MA, July 2000.

[60] M. Harchol-Balter. Process lifetimes are not exponential, more like $1/t$: implications on dynamic load balancing. Technical report, EECS, University of California, Berkeley, 1996. CSD-94-826.

[61] M. Harchol-Balter and A. Downy. Exploiting process lifetime distributions for dynamic load balancing. In *Proc. SIGMETRICS '96*, pages 13-24, 1996.

[62] B. Hari, R. Hariharan, S. Srinivasan. An integrated congestion management architecture for Internet hosts. In *Proc. ACM SIGCOMM '99*, September 1999.

[63] E. Hashem. Analysis of random drop for gateway congestion control. MIT Technical Report, 1990.

[64] M. Hayes. Statistical digital signal processing and modeling. New York:Wiley, 1996.

[65] D. Heath, S. Resnick, and G. Samorodnitsky. Heavy tails and long range dependence in on/off processes and associated fluid models. In *Mathematics of Operations Research*, Vol. 23, pages 145-165, 1998.

[66] N. Hohn and D. Veitch. Inverting sampled traffic. In *Proc. of ACM IMC '03*, Florida, USA, October 2003.

[67] C. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proc. of IEEE INFOCOM '01*, April 2001.

[68] J. R. M. Hosking. Fractional differencing. Biometrika, 68, pages 165-176, 1981.

[69] Information networks division, Hewlett-Packard Company. Netperf: A network performance benchmark, http://www.netperf.org, 1995.

[70] Internet Protocol Flow Information eXport (IPFIX). IETF Working Group, *http://ipfix.doit.wisc.edu*.

[71] IPMA: Internet performance measurement and analysis. For more information, refer to http://www.merit.edu/ipma.

[72] http://www.ircache.net.

[73] V. Jacobson. Presentations to the IETF performance and congestion control working group. August 1989.

[74] V. Jacobson. Pathchar — a tool to infer characteristics of Internet paths. Available at ftp://ftp.ee.lbl.gov/pathchar/, 1997.

[75] P. Jacquet. Analytic information theory in service of queueing with aggregated exponential on/off arrivals. In *Proc. 25th Allerton Conference on Communication, Control and Computing*, pages 242-251, 1996.

[76] M. Krunz, A. Makowski. A source model for VBR video traffic based on M/G/infinity input process. In *Proc. of IEEE INFOCOM '98*, San Francisco, CA, April 1998.

[77] S. Kunniyur and R. Srikant. Analysis and design of an Adaptive Virtual Queue (AVQ) algorithm for active queue management. In *Proc. of ACM SIGCOMM '01*, August 2001.

[78] K. Lai and M. Baker. Measuring bandwidth. In *Proc. of IEEE INFOCOM '99*, April 1999.

[79] K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proc. of ACM SIGCOMM '00*, August 2000.

[80] W. E. Leland and T. J. Ott. UNIX process behavior and load balancing among loosely-coupled computers. In O. J. Boxma, J. W. Cohen, and H. C. Tijims, eds, *Teletraffic Analysis and Computer Performance Evaluation*, pages 191-208, Elsevier, Amsterdam, The Netherlands, 1986.

[81] W. E. Leland, and D. V. Wilson. High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In *Proc. of IEEE INFOCOM '91*, pages 1360-1366, Bal Harbour, FL, 1991.

[82] W. E. Leland, M. S. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic. In *Proc. of ACM SIGCOMM '93*, pages 183-193, 1993.

[83] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson. On the self-similar nature of Ethernet traffic (Extended Version). *IEEE/ACM Transactions on Networking*, February 1994.

[84] N. Likhanov, B. Tsybakov, and N. Georganas. Analysis of an ATM buffer with self-similar ("fractal") input traffic. In *Proc. of IEEE INFOCOM '95*, pages 985-992, 1995.

[85] J. S. Lim and A. V. Oppenheim. Advanced topics in signal processing. *ch.4. Prentice-Hall*, 1988.

[86] W. Lin, and R. Morris. Dynamics of Random Early Detection. In *Proc. of ACM SIGCOMM '97*, September 1997.

[87] N. X. Liu and J. S. Baras. Statistical modeling and performance analysis of multi-scale traffic. In *Proc. of ACM INFOCOM '03*, San Francisco, April 2003.

[88] *http://cm.bell-labs.com/cm/ms/departments/sia/InternetTraffic/S-Net/.*

[89] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson and S. Shenker. Controlling high bandwidth aggregates in the network. http://www.aciri.org/pushback/, July 2001.

[90] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis. Creating a scalable architecture for Internet measurement. In *Proc. INET '98,* 1998.

[91] A. Mankin. Random drop congestion control. In *Proc. of ACM SIGCOMM '90*, pages 1-7, September 1990.

[92] M. Mathis, J. Semkeand, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm, *Computer Communication Review*, Vol. 27, pages 67-82, July 1997.

[93] K. Maulik and S. Resnck. Small and large time scale analysis of a network traffic model. In *Mathematics Subject Classification 1991*, 1991.

[94] V. M. Misra, W. B. Gong. A hierachical model for teletraffic. In *proc. 37th IEEE Conf. Decision and Control*, Vol. 2, pages 1674-1679, 1998.

[95] R. J. Mondragon, J. M. Pitts, D. K. Arrowsmith. Chaotic intermittency-sawtooth map model of aggregated self-similar traffic streams. *Electronic Letters*, Vol. 36, No. 2, pages 184-186, 2000.

[96] I. Norros. On the use of Fractional Brownian Motion in the theory of connectionless networks. *IEEE Journal on selected areas in communications*, Vol. 13, No. 6, August 1995.

[97] I. Norros. A storage model with self-similar input. *Queueing Systems*, Vol. 16, pages 387-396, 1994.

[98] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proc. of IEEE INFOCOM '99*, New York, USA, March 1999.

[99] K. Park, G. T. Kim, M. E. Crovella. On the relationship between files sizes, transport protocols, and self-similar network traffic. In *Proc. of the Fourth International Conference on Network Protocols (ICNP '96)*, pages 171-180, October 1996.

[100] K. Park, W. Willinger, Self-similar network traffic and performance evaluation. Ch. 20, Wiley-Interscience.

[101] M. Parulekar and A. Makowski. Tail probabilities for a multiplexer with self-similar traffic. In *Proc. of IEEE INFOCOM '96*, pages 1452-1459, 1996.

[102] Packet sampling (PASAMP). IETF working group, *http://ops.ietf.org/psamp/*.

[103] V. Paxson, S.Floyd. The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, Vol. 3, No. 3, pages 226-244, June 1995.

[104] V. Paxson. End-to-end Internet packet dynamics. In *Proc. of ACM SIGCOMM '97*, September 1997.

[105] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Trans. Networking*, Vol. 7, pages 277-292, June 1999.

[106] S. Ratnasamy and S. Mccanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proc. of IEEE INFOCOM '99*, March 1999.

[107] S. Resnick. Adventures in stochastic processes. Birkhauser, Boston, 1992.

[108] V. Ribeiro, M. S. Crouse, R. Riedi, and R. G. Baraniuk. Simulation of non-Gaussian long-range-dependent traffic using wavelets. In *Proc. of SIGMETRICS '99*, Vol. 27, No. 1, pages 1-12, June 1999.

[109] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk. Multifractal cross-traffic estimation. In *Proc. of the ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management,* September 2000.

[110] R. Riedi, M. S. Crouse, V. Ribeiro, and R. G. Raraniuk. A multifractal wavelet model with application to TCP network traffic. IEEE Trans. Inf. Theory, Special Issue on "Multiscale Statistical Signal Analysis and Its Applications," Vol. 45, No. 3, pages 992-1018, April 1999.

[111] Roughan, Veitch and Abry. Real-time estimation of the parameters of long-range dependence (extended version). *IEEE/ACM Transactions on Networking*, Vol. 8, No. 4, pages 467-478, August 2000.

[112] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. In *Proc. of ACM SIGMETRICS '00*, September 2000.

[113] A. RubiNi and J. Corbet Linux Device Drivers. *O'Reilly & Associates, Inc.*,2001.

[114] A. Sang, and S. Li. A predictability analysis of network traffic. In *Proc. of IEEE INFOCOM '00*, Tel-Aviv, Israel, March 2000.

[115] S. Savage. Sting: A TCP-based network measurement tool. In *Proc. of the USENIX Symp. on Internet Technologies and Systems*, 1999.

[116] A. Shaikh, J. Rexford, and K. Shin. Load-sensitive routing of long-lived IP flows. In *Proc. of ACM SIGCOMM '99*, pages 215-226, 1999.

[117] B. Sikdar, K. S. Vastola. The effect of TCP on the self-similarity of network traffic. 2001 Conference on Information Sciences and Systems, The Johns Hopkins University, March 2001.

[118] Surveyor. For more information, refer to http://io.advanced.org/surveyor.

[119] M. S. Taqqu, W. Willinger, R. Sherman. Proof of a fundamental result in self-similar traffic modeling. In *ACM SIGCOMM Computer Communication Review*, Vol. 27, No. 2, pages 5-23, April 1997.

[120] K. Thompson, G. J. Miller and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, November 1997.

[121] B. Tsybakov and N. Georganas. On self-similar in ATM queues: definitions, overflow probability bound and cell delay distributions. *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3, pages 397-409, June 1997.

[122] B. Tsybakov and N. D. Georganas. Self-similar traffic and upper bounds to buffer overflow in an ATM queue. *Performance Evaluation*, Vol. 36, No. 1, pages 57-80, 1998.

[123] T. Tuan and K. Park. Multiple time scale congestion control for self-similar network traffic. *Performance Evaluation*, Vol. 36, pages 359–386, 1999.

[124] T. Tuan and K. Park. Multiple time scale redundancy control for QoS-sensitive transport of real-time traffic. In *Proc. of IEEE INFOCOM '00*, April 2000.

[125] UCB, LBNL, VINT Network Simulator. *http://www-mash.cs.berkeley.edu/ns/.*

[126] J. L. Vehel. Fractal and multifractal Internet traffic. In *Business Briefing: Global Optical Communications*, 2002.

[127] A. Veres, Z. Kenesi, S. Molnar, and G. Vattay. On the propagation of Long-Range Dependence in the Internet. In *Proc. of ACM SIGCOMM '00*, Stockholm, Sweden, August 2000.

[128] J. Walrand and P. Varaiya. High-performance communication networks. Morgan Kaufmann Publishers, Inc. San Franciso, California, 1996.

[129] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. In *Proc. of ACM SIGCOMM '95*, pages 100-113, 1995.

[130] W. Willinger, V. Paxson, M. S. Taqqu. Self-similarity and heavey-tails: structural modeling of network traffic. In *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, R.J. Adler, R.E¿ Feldmann and M.S. Taqqu, editors. ISBN 0-8176-3951-9, Birkhauser, Boston, 1998.

[131] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. In *Proc. of ACM SIGCOMM '91*, pages 149-157, 1991.

[132] G. R. Wright and W. R. Stevens. TCP/IP Illustrated, Volume 2: The Implementation. Addison Wesley Publishing Company, 1995.

[133] T. Yoshihara, S. Kasahara, Y. Takahashi. Practical time-scale fitting of self-similar traffic with Markov-modulated Passion process. *Telecommunication Systems*, Vol. 17, No.1-2, pages 185-211, 2001.

[134] M. Zukerman, T. D. Neame, R. G. Addie. Internet traffic modeling and future technology implications. In *Proc. of ACM INFOCOM '03*, San Francisco, April 2003.

Vita

Guanghui He was born in XinJiang Province, China. He received his M.S. and B.S. in Electrical Engineering in May 1996 and June 1993 at Tsinghua University, P.R. China. He joined Department of Computer Science, University of Illinois at Urbana-Champaign in 2001. In his Ph.D. research work, he proposed a hierarchical model for the Internet traffic and the scaling properties of the Internet traffic are exploited to facilitate resource and traffic management (active queue management and congestion control). He also exploited the self-similarity of the Internet traffic for the sake of network end-to-end measurement and traffic sampling.